UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Diseño, implementación y control de un prototipo de vehículo de Ackermann

Trabajo Fin de Máster

Máster Universitario en Ingeniería Mecatrónica

AUTOR/A: Johannesen , Sindre

Tutor/a: Casanova Calvo, Vicente Fermín

CURSO ACADÉMICO: 2021/2022

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

mim máster en ingeniería mecatrónica

Diseño, Implementación y control de un prototipo de vehículo de Ackermann

MEMORIA PRESENTADA POR:

Sindre Johannesen

Máster Universitario en Ingeniería Mecatrónica

DIRECTOR:

Casanova Calvo, Vicente Fermín

Julio 2022

ÍNDICE

1 Introducción

En este primer apartado, se pretende aportarle al lector una idea general respecto del objetivo del trabajo realizado así como dar un breve resumen del contenido de la memoria. Se definirán claramente cuáles fueron los objetivos que se pretendían lograr con la realización de este trabajo, así como el motivo de que se haya decidido hacer el mismo.

El trabajo incluye el diseño, simulación y montaje de un prototipo real de un vehículo de Ackermann, así como varias simulaciones del mismo que fueron realizadas antes de montar el prototipo.

1.1 Resumen del proyecto

El proyecto realizado trata, como bien indica su propio título, sobre el diseño, implementación y control de un vehículo de Ackermann. Esto incluye el diseño, simulación y montaje del propio vehículo, así como la programación del sistema de control una vez montado el prototipo.

Este tipo de vehículo dispone de 4 ruedas dispuestas de manera que las 2 ruedas delanteras sean capaces de girar sobre un eje perpendicular al suelo, mientras que las 2 ruedas traseras solo son capaces de girar respecto del eje que las une. Esta idea se puede apreciar en la imagen inferior, en el que las ruedas delanteras del vehículo, las cuales denominaremos ruedas directrices, giran un ángulo α y β, mientras que las ruedas traseras se mantienen paralelas a la dirección de movimiento del vehículo.

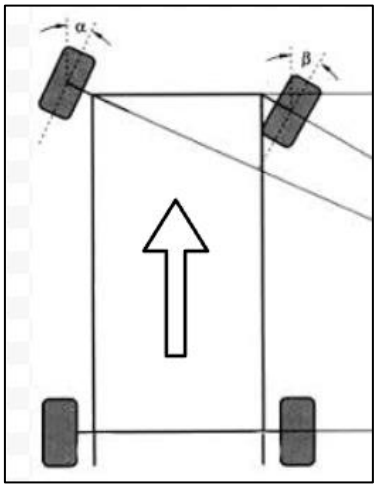


Ilustración 1: Vehículo

El vehículo se denomina 'Vehículo de Ackermann' ya que su sistema de dirección debe presentar una disposición geométrica llamada 'Geometría de Ackermann'. Este sistema de dirección es muy similar al sistema de dirección que presenta un automóvil real.

El motivo de que se requiera diseñar un mecanismo de dirección que cumpla dicha geometría, se debe a que al girar ambas ruedas delanteras un mismo ángulo, estas generarían unas trayectorias curvas cuyos radios de giro son de distinto tamaño. Esto provocaría un comportamiento inestable del vehículo al girar así como un aumento innecesario en el desgaste de las ruedas.

La geometría de Ackermann resuelve este problema diseñando el mecanismo de dirección como un paralelogramo con unas dimensiones y características especiales que describiremos más detalladamente en el capítulo 2.

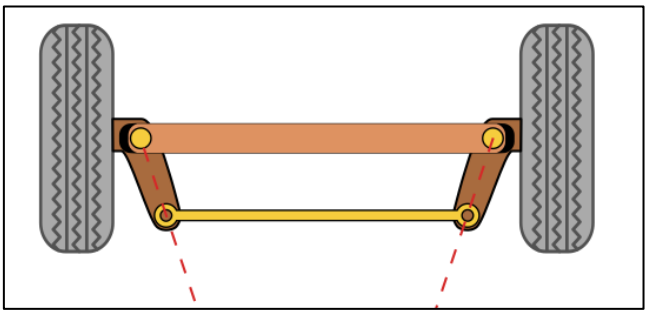


Ilustración 2: Paralelogramo de dirección

Esta disposición geométrica del mecanismo de dirección provoca que, al girar las ruedas directrices, ambas giren en torno a un mismo punto central que llamaremos 'centro de giro'.

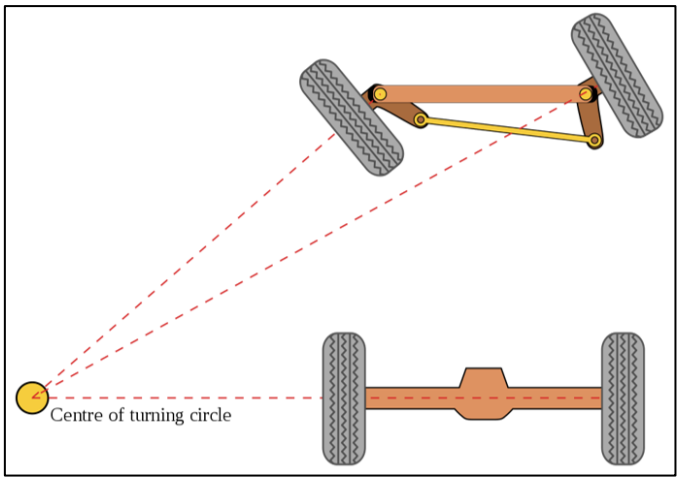


Ilustración 3: Giro vehículo Ackermann

Con esto se logra reducir el desgaste producido en las ruedas al girar el vehículo, así como aportarle una mayor estabilidad al vehículo durante los giros.

Se ha realizado el diseño completo del vehículo, desde el propio diseño mecánico del mismo, aprovechando tanto componentes comerciales como componentes que hayan sido creados mediante impresión 3D o corte por láser, hasta el diseño del sistema de control que dirige al vehículo.

Durante la fase de simulación, se ha diseñado además un sistema de control de trayectoria que permite que el vehículo siga una trayectoria predefinida. Mediante esta parte de la simulación se logra comprobar que el vehículo diseñado se encuentra lo suficientemente bien diseñado

como para realizar una tarea compleja tal y como es el seguimiento automático de trayectorias. Cabe mencionar que debido a que este vehículo no permite desplazarse en cualquier dirección y su radio de giro vendrá limitado por la construcción del propio mecanismo de dirección, la trayectoria a seguir deberá presentar unas características especiales que definiremos en el capítulo de simulación.

Debido a la complejidad y envergadura de un trabajo de esta índole, se ha optado por inicialmente crear un diseño 3D y posteriormente simular dicho diseño en un programa de simulación mecánica. Las herramientas informáticas que se han utilizado para realizar dichas tareas son:

- Solidworks: Para realizar el propio diseño del vehículo.
- Matlab/Simulink: Para simular el sistema de control diseñado para el vehículo.

El motivo de que se realice la simulación del vehículo se debe principalmente a que esto nos permite comprobar por una parte que el diseño mecánico del vehículo es adecuado para la aplicación deseada, así como para comprobar el correcto funcionamiento del sistema de control diseñado para controlar el vehículo. En la imagen inferior se puede apreciar el aspecto del vehículo durante la simulación así como una de las trayectorias que sería capaz de seguir el vehículo:

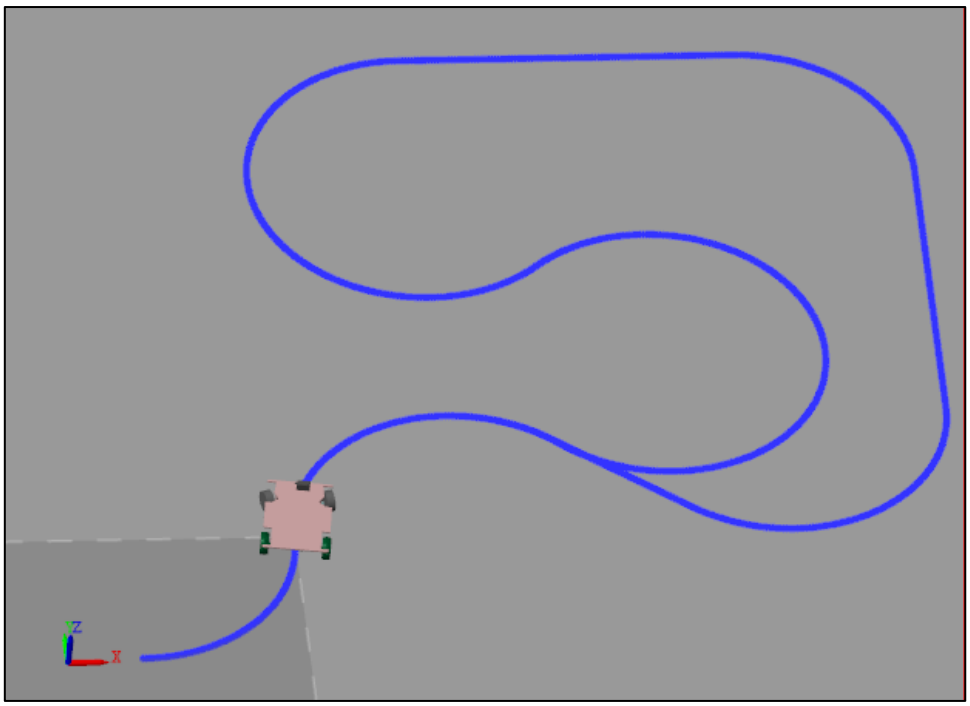


Ilustración 4: Simulación vehículo

En este entorno de simulación se han diseñado los sistemas de control del vehículo, entre los que se incluye el sistema de control de trayectoria y un controlador PI de velocidad que controla la velocidad a la que se desplaza el vehículo.

Tras realizar las pertinentes simulaciones del vehículo se procede a montar el prototipo real mediante piezas tanto comerciales como creadas en el laboratorio de prototipado de la ETSID.

El Prototipo real que se construyo se puede apreciar en la imagen inferior:

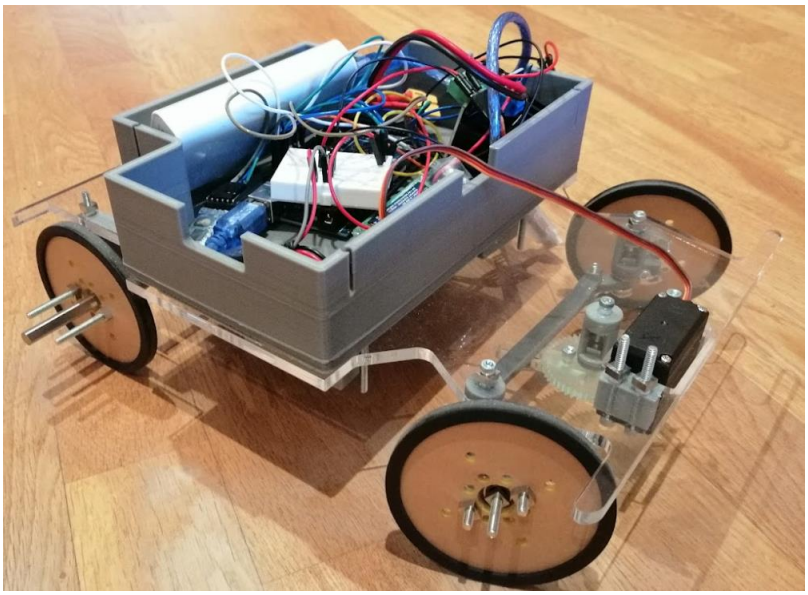


Ilustración 5: Prototipo Montado

Debido a la falta de precisión en la construcción del mecanismo de dirección, este prototipo tendría bastantes dificultades en utilizarse para realizar un seguimiento de trayectoria.

Sin embargo, se ha logrado diseñar un sistema de control manual que permite controlar el avance/retroceso del vehículo, así como el giro del vehículo. Este control manual se programó de forma que pudiese realizarse tanto con un mando de videoconsola como con un smartphone que disponga de conexión Bluetooth. El control mediante smartphone permite además realizar un control avanzado sobre la velocidad del vehículo ya que nos permite controlar los valores característicos del controlador PI así como visualizar la velocidad actual del vehículo junto a su referencia.

1.2 Objetivos

El principal objetivo de este trabajo es el de diseñar un vehículo con un sistema de dirección formado por el Mecanismo de Ackermann. Esto nos permitiría aprender durante el proceso de diseño sobre el funcionamiento del mecanismo de dirección de un vehículo real.

Debido a que el vehículo diseñado se asemeja a un vehículo real, este nos podría servir para testear sistemas de dirección autónoma pensados para vehículos reales, un tema de gran relevancia en la actualidad.

Como todo trabajo académico, uno de los objetivos principales es aprender conceptos nuevos. En este caso se han aprendido sobre conceptos de simulación de vehículos mediante la herramienta Simulink Simscape, así como conceptos relativos al diseño mecánico de un sistema de dirección.

Finalmente se tratará de destacar las dificultades que pueden aparecer en el proceso de montaje de un prototipo real con respecto a un modelo que ha sido generado mediante un software CAD.

El motivo de que se haya escogido este trabajo es en gran parte debido a su elevada relevancia en la actualidad en la que los vehículos de conducción autónoma son cada vez más comunes. Resulta además un proyecto que abarca un amplio abanico del campo de la mecatrónica desde el diseño mecánico del vehículo hasta la propia programación del mismo.

1.3 Descripción de la memoria

La memoria se encuentra estructurada en un total de 5 apartados y 3 Anexos. Al inicio de cada apartado se realiza una breve introducción al contenido del mismo. A pesar de ello trataremos de proporcionar a continuación los conceptos clave que se tratarán en cada apartado.

En el primer apartado, en el cual nos encontramos actualmente, se aporta un resumen del trabajo realizado así como una primera introducción al mecanismo de Ackermann, tema principal del documento.

En el segundo capítulo trataremos de definir los conceptos teóricos que son necesarios para poder entender claramente los conceptos tratados en el resto del documento. El capítulo comienza con una primera introducción a los tipos de vehículos de 4 ruedas que suelen ser habituales en el campo de la robótica, seguido de una explicación teórica de los conceptos de cinemática que intervendrán en el vehículo que se pretende diseñar.

Seguidamente se pasa a detallar todas las características geométricas que definen al mecanismo de Ackermann, incluyendo las fórmulas matemáticas que usaremos para obtener las dimensiones del sistema de dirección.

Se dará una introducción teórica a los sistemas de control que se pretende diseñar para el vehículo. Estos serán un controlador PI, que usaremos para controlar la velocidad del motor del vehículo y un control de seguimiento de trayectoria con la que se pretende que el vehículo sea capaz de seguir una trayectoria predefinida.

En el capítulo 3 se incluye la descripción del modelo diseñado para realizar la simulación del vehículo que se pretende posteriormente montar. Se incluyen además los resultados obtenidos tras simular el vehículo en varias condiciones en función del nivel de control que se le aporte al mismo, es por esto por lo que abarcará la mayor parte de la memoria.

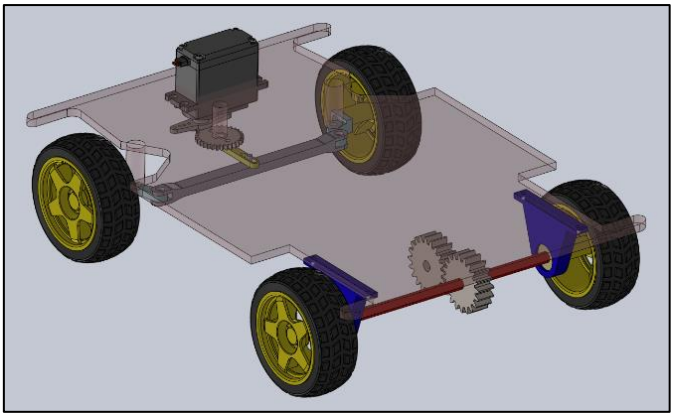


Ilustración 6: Prototipo Inicial

Se incluye primeramente una descripción del vehículo, dividiendo la descripción del mismo en 2 apartados:

- Sistema de tracción
- Sistema de dirección

Una vez hayamos descrito el propio vehículo pasaremos a describir los dos sistemas de control que se han diseñado para el vehículo. Un sistema de control manual y un sistema de control autónomo que permite que el vehículo siga una trayectoria predefinida sin intervención humana.

Tras definir dichos sistemas de control, se detalla el modelo realizado en Simulink para simular el comportamiento que cabría esperar del vehículo diseñado. Se describe detalladamente el modelo y se comenta como se ha implementado el sistema de control autónomo, así como los resultados obtenidos para dicha simulación.

En el capítulo 5 veremos el prototipo real que se ha decidido montar junto a una descripción de todos sus componentes. También se incluye una breve descripción del conexionado eléctrico del vehículo así como la manera en la que se ha implementado el controlador PI de velocidad del vehículo. El control del avance y giro del vehículo se puede realizar mediante un mando de videoconsola o mediante un smartphone con conexión Bluetooth, pero el smartphone permite un mayor rango de funcionalidades.

En la siguiente imagen puede apreciarse como aprovechando una aplicación móvil, pudimos implementar un controlador de velocidad que nos permite modificar las variables Kp y KI del controlador PI de forma inalámbrica, así como mostrar la velocidad del vehículo junto a la referencia aplicada:

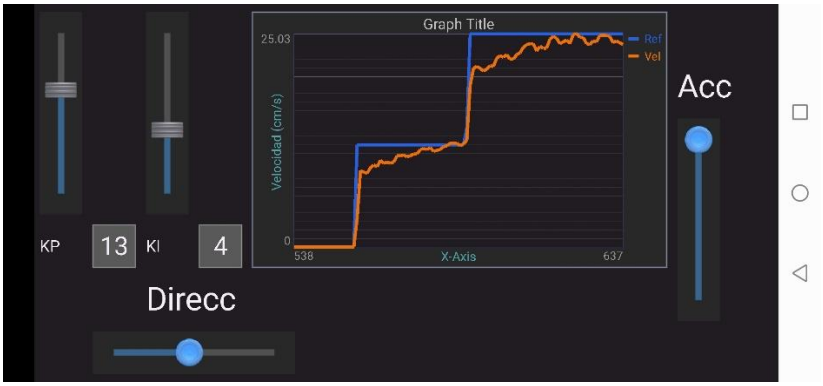


Ilustración 7: Control por Smartphone

Al final del capitulo se incluye un apartado en el que se describen los distintos problemas que aparecieron durante el montaje del vehículo y los pasos que fueron necesarios tomar para resolver dichos problemas.

Al final de la memoria se han incluido unos Anexos que referencian elementos que, debido a su longitud, no se han podido incluir en el resto de la memoria:

Anexo I: Planos

Anexo II: Códigos realizados

Anexo III: Datasheet

2 Conceptos teóricos y estudio previo

En este capítulo se tratará de describir los distintos conceptos teóricos que el lector requerirá conocer para poder comprender adecuadamente el contenido de la memoria así como una introducción más exhaustiva a la geometría de Ackermann.

Se comenzará el capítulo dando una introducción a los vehículos de 4 ruedas debido a que es el tema principal de la memoria y conviene que el lector conozca los distintos tipos de vehículos que suelen ser habituales en este tipo de trabajos. Se incluyen además conceptos básicos de cinemática y diseño de controladores automáticos ya que se considera indispensable para comprender la totalidad del trabajo realizado.

Debido a que para realizar el control de seguimiento del vehículo se ha requerido usar una modelización del vehículo de 4 ruedas, se ha incluido un apartado separado describiendo la modelización realizada, además de la propia descripción del sistema de seguimiento.

2.1 Vehículo de 4 ruedas

Debido a que el trabajo trata sobre el diseño y montaje de un vehículo de 4 ruedas, daremos una descripción detallada sobre los detalles constructivos básicos de este tipo de vehículo y las distintas formas de manejar este tipo de vehículo. Se describirá además 3 tipos de construcción de este tipo de vehículos utilizados comúnmente en el campo de la robótica.

Como bien indica su nombre, un vehículo de 4 ruedas, se caracteriza por disponer de 4 ruedas además de 1 o más motores que propulsan dichas ruedas para desplazar el vehículo sobre el que se encuentren montados. El caso más común siendo un automóvil, utilizado principalmente para transporte de pasajeros.

Ilustración 8: Vehículo automóvil

Vehículo de Ackermann

Este tipo de vehículo coincide con el tipo de vehículo que se pretende diseñar, sobre el cual se ha realizado ya una pequeña introducción.

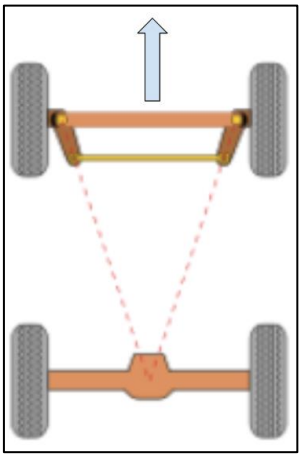


Ilustración 9: Vehículo de Ackermann

Se trata de un vehículo montado de forma que las 2 ruedas traseras se encuentren sobre un mismo eje paralelo al suelo y perpendicular a la dirección de desplazamiento del vehículo, mientras que las 2 ruedas delanteras (ruedas directrices) giran independientemente sobre su propio eje como podemos ver en la imagen inferior. En este tipo de montajes es habitual que las ruedas traseras actúen como la fuerza motriz del sistema, ya sea mediante un único motor conectado al eje trasero, o un motor conectado a cada rueda trasera, funcionando de forma simultánea. Las ruedas delanteras podrán girar conjuntamente sobre un eje perpendicular a su eje de giro, permitiendo con ello que el vehículo se desplace en una dirección u otra.

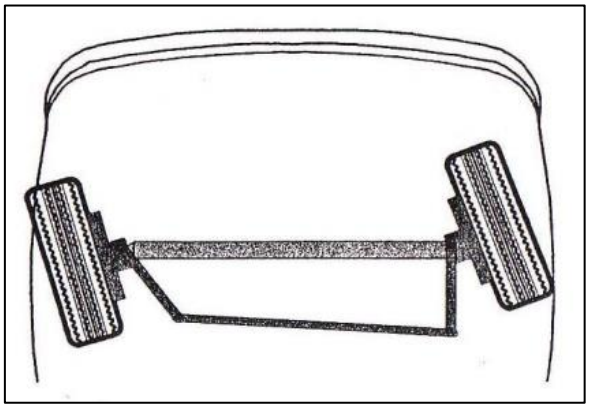


Ilustración 10: Giro mecanismo de Ackermann

Este tipo de vehículo se caracteriza por poder realizar movimientos de avance y retroceso en una sola dirección, a diferencia de los otros dos modelos de vehículos que veremos a continuación. Para ello el vehículo se impulsa mediante las ruedas traseras, y las ruedas delanteras dirigen el movimiento del mismo.

El movimiento que produce este tipo de vehículo al realizar un giro resulta un círculo con un radio R, cuyo valor vendrá definido por el mecanismo de dirección así como el tamaño del propio vehículo.

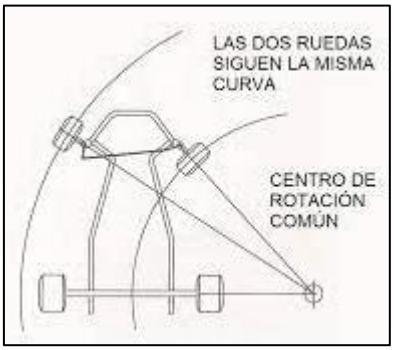


Ilustración 11: Desplazamiento vehículo de Ackermann

Vehículo de 4 motores (4WD)

El que probablemente sea el vehículo más popular en el mundo de la robótica, es el vehículo compuesto por 4 ruedas cada una de ellas conectadas de forma independientemente a su propio motor. En la imagen inferior se puede observar una disposición habitual de las ruedas para este tipo de vehículos.

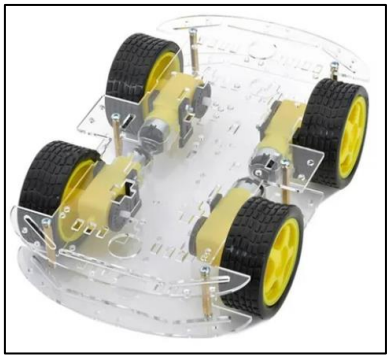


Ilustración 12: Vehículo 4WD

En esta disposición se debe controlar el giro de las 4 ruedas de forma simultánea para lograr el movimiento deseado. Para ello se debe tener en cuenta, que las 2 ruedas que se encuentran en un mismo lateral del vehículo deben de girar en la misma dirección, de otra forma el vehículo no sería capaz de desplazarse.

Este tipo de vehículo permite un mayor rango de movimientos que el vehículo anterior al poder girar de forma independiente las 4 ruedas que lo componen.

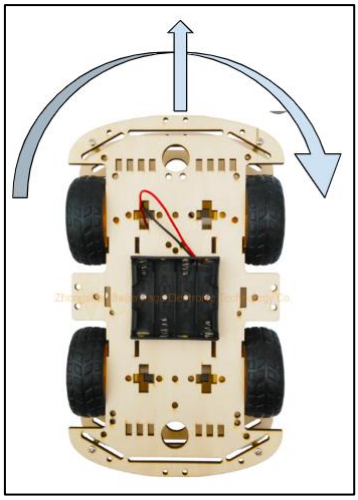


Ilustración 13: Movimiento 4WD

El movimiento adicional que permite este vehículo con respecto al anterior es la capacidad de girar respecto a un eje perpendicular al propio vehículo, es decir, es capaz de girar sobre si mismo para orientarse de cualquier forma sin necesidad de avanzar.

Para lograr este giro, es necesario que las ruedas de cada lateral giren en sentidos opuestos. En la siguiente imagen se aprecia el giro que tendrían que producir las ruedas para que el vehículo gire de forma horaria con respecto a un eje perpendicular al suelo que pase por su centro geométrico.

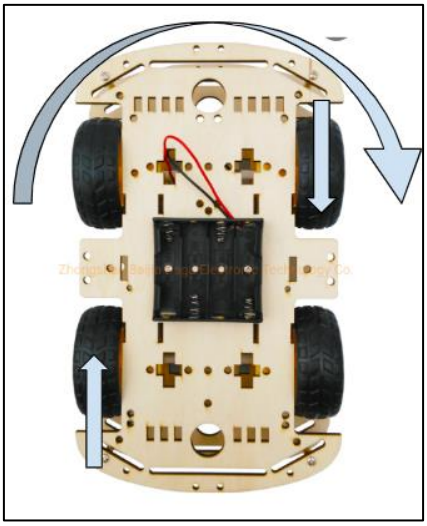


Ilustración 14: Giro vehículo 4WD

Como puede apreciarse las 2 ruedas laterales derechas tratarán de que el vehículo retroceda, mientras que las 2 ruedas laterales izquierdas tratarán de que el vehículo avance. Esto provoca el giro descrito anteriormente. Para evitar que las ruedas deslicen, es imperativo que todas las ruedas giren a la misma velocidad angular.

Vehículo omnidireccional

Este tipo de vehículo tiene la misma geometría constructiva que el vehículo anterior, 4 ruedas cada una con un motor independiente.

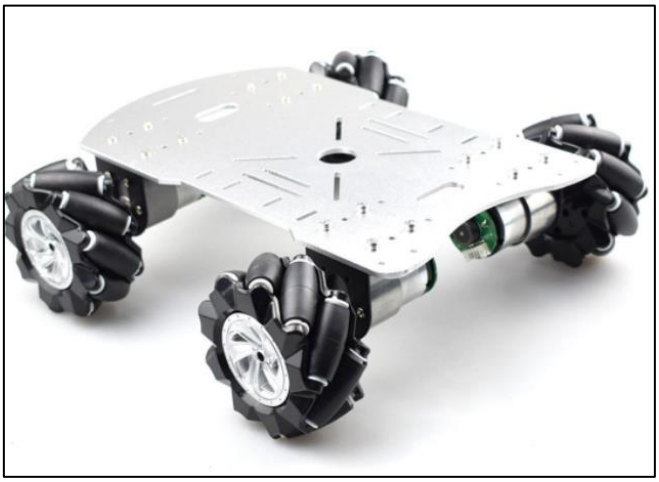


Ilustración 15: Vehículo omnidireccional

La diferencia de este vehículo con el anterior, se encuentra en las ruedas que dispone el vehículo. Estas ruedas disponen de varios rodillos montados sobre la propia estructura de la rueda, capaces de girar sobre si mismos. Gracias a este tipo de rodillos el vehículo es capaz de desplazarse lateralmente, un movimiento que los otros vehículos no eran capaces de realizar.

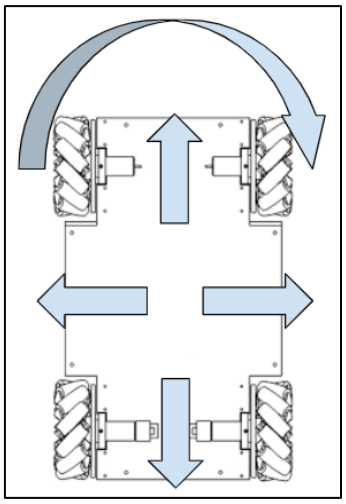


Ilustración 16: Movimiento omnidireccional.

En la imagen anterior se pueden apreciar todos los movimientos que es capaz de realizar el vehículo. La dirección que tenga el vehículo dependerá de la dirección del giro que se le aplique a las 4 ruedas.

2.2 Modelización vehículo

A la hora de realizar cualquier sistema de control para un sistema, conviene tratar de realizar un modelo simplificado del mismo para facilitar el diseño del sistema de control que se pretende diseñar. Para el caso de un vehículo de Ackermann es habitual representar el vehículo de 4 ruedas como una bicicleta de 2 ruedas como se aprecia en la siguiente imagen:

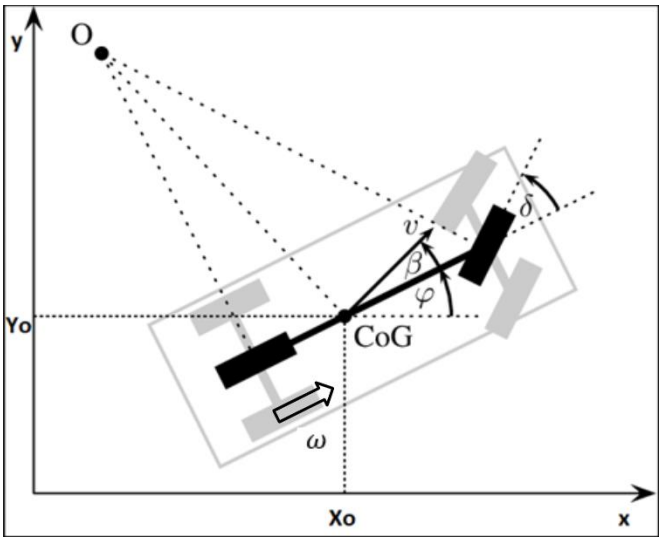


Ilustración 17: Modelo Simplificado Vehículo

En este sistema se modelizan las 4 ruedas que dispone el vehículo como 2 ruedas unidas mediante una barra: una rueda trasera que siempre irá alineada con el resto del vehículo y una rueda delantera capaz de girar sobre un eje perpendicular al suelo, funcionando como única rueda directriz. Ambas ruedas pasan por el centro del vehículo como vemos en la imagen anterior.

Las variables características que usaremos para el sistema de control del vehículo son:

- *Xo* e *Yo*: Coordenadas del vehículo
- φ*:* Ángulo de giro del propio vehículo con respecto de la horizontal
- δ: Ángulo de giro de la rueda directriz mediante un ángulo (esta será nuestra variable de acción de control al diseñar el sistema de dirección autónoma)
- β: ángulo que forma el vector de velocidad con la horizontal
- ω: Velocidad angular de las ruedas del vehículo

Cabe mencionar que en función del método de control que se esté aplicando, las coordenadas *Xo* e *Yo* pueden referirse al centro geométrico del vehículo, al centro de masa del mismo (que no necesariamente coincidirá con el centro geométrico del mismo) o incluso al punto central del eje trasero, que para nuestro modelo simplificado correspondería con el centro de apoyo de la rueda trasera de nuestro modelo.

A la hora de diseñar el sistema de control lo consideraremos como un sistema en el que se dispone de 2 entradas de control sobre la que podremos actuar:

- Voltaje aplicado al motor eléctrico
- Ángulo del servomotor.

Actuando sobre estas dos variables el vehículo se comportará de una manera u otra. Dicho comportamiento lo mediremos mediante 2 variables de salida:

- Velocidad del vehículo
- Orientación del vehículo con respecto a la horizontal

Con esta modelización resulta sencillo obtener el ángulo de giro que requieren las ruedas del vehículo para realizar un giro de radio R:

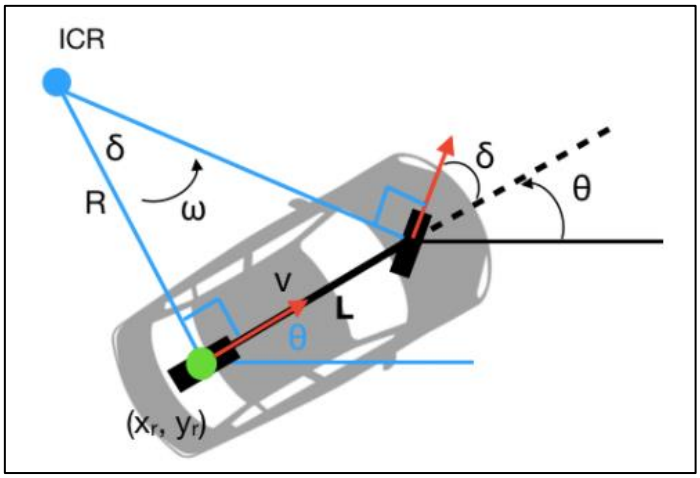


Ilustración 18: Radio de giro - Angulo giro rueda

ICR denomina el centro instantáneo de rotación, describiendo el punto alrededor del cual gira el vehículo. De la ilustración se observa que el radio de giro que se obtiene para un ángulo de giro δ de la rueda directriz se puede obtener con la siguiente fórmula:

$$R = \frac{L}{\tan(\delta)}$$

A efectos prácticos, no se tomará en cuenta el balanceo ni la altura a la que se encuentre el vehículo, limitando con ello el estudio a un solo plano X-Y por el que se desplazará el vehículo.

Debido a que el vehículo que se pretende diseñar tiene un peso ligero y no alcanzará velocidades elevadas, no entraremos en el estudio dinámico del mismo, limitando este trabajo al estudio de la cinemática y dirección del vehículo diseñado.

2.3 Cinemática

En este capítulo se describirán los conceptos relacionados con la cinemática que serán relevantes para comprender el trabajo. Se comenzará primero describiendo la cinemática propia de la rueda del vehículo diseñado, seguido de un modelo cinemático del vehículo completo.

Cinemática de una rueda

La velocidad angular de las ruedas indica el número de rotaciones que una rueda realiza sobre su eje en un tiempo determinado. Suele denotarse por el símbolo 'ω' o mediante la letra 'n'. Este valor suele medirse en radianes por segundo (rad/s) o revoluciones por minuto (rpm). La relación entre estas 2 unidades de medida es la siguiente:

$$1rpm = \frac{2\pi}{60} rad/s$$

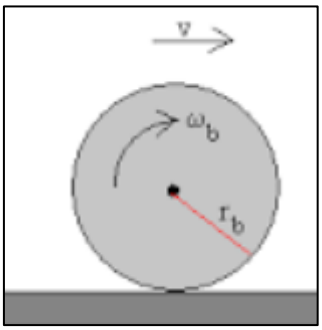


Ilustración 19: Rueda en movimiento

La relación entre la velocidad lineal a la que se desplaza una rueda, y la velocidad angular de la misma dependerá del tamaño de la propia rueda, estando esta definida por su radio 'r'. La siguiente fórmula nos permitirá obtener la velocidad lineal de una rueda en función de su velocidad angular y con ello la velocidad lineal del vehículo al que se encuentra acoplado:

$$v(m/s) = \omega(rad/s) \cdot r(m)$$

Mediante esta fórmula podremos obtener fácilmente la velocidad lineal de la rueda del vehículo, siempre y cuando está se desplace en dirección perpendicular al giro de la misma.

Conviene mencionar que esta fórmula solo es aplicable para el caso en el que las ruedas del vehículo giren sobre el suelo sin deslizarse. En caso contrario requeriremos de instrumentación más compleja para obtener la velocidad real del vehículo.

Cinemática vehículo de Ackermann

La velocidad lineal del vehículo completo, coincidirá con la velocidad de cualquiera de sus ruedas si se desplaza en línea recta sin ningún ángulo de giro. En el momento en el que las ruedas directrices comiencen a girar, aparecerá una componente tangencial de velocidad que conviene tomar en consideración.

Si partimos del modelo descrito en el apartado anterior, podremos representar el vehículo mediante las siguientes variables:

- x, y: Coordenadas del eje trasero del vehículo
- θ: Ángulo de giro del vehículo con respecto al eje X
- δ: Ángulo de giro de la rueda del vehículo

Las entradas que actúan sobre dichas variables van a ser la velocidad lineal del vehículo, la cual podremos controlar en función del voltaje aplicado al motor, y el ángulo de giro de las ruedas del vehículo.

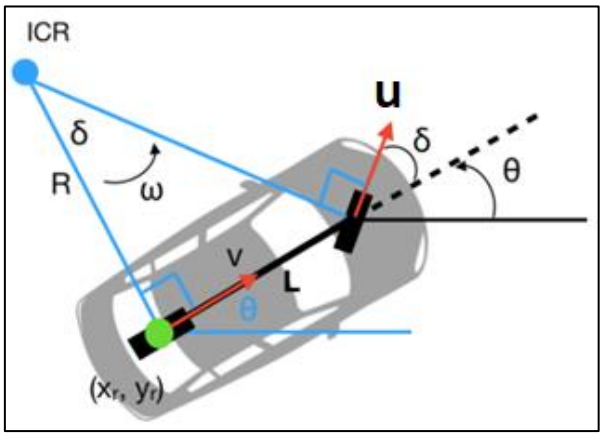


Ilustración 20: Cinemática Ackermann

Definiremos la variable *v* como la velocidad lineal del vehículo, la cual podremos obtener midiendo directamente la velocidad de las ruedas traseras del vehículo, asumiendo que no se produzca ningún deslizamiento. Definiremos además una variable u, que usaremos para representar la velocidad de la rueda frontal del vehículo, sobre la cual se calcula la velocidad angular del vehículo.

De estas variables podemos obtener las ecuaciones que representan la velocidad lineal del vehículo en el espacio como:

- $\dot{x} = v \cdot \cos(\theta)$
- $\dot{y} = v \cdot \sin(\theta)$

Para poder obtener la velocidad de giro del vehículo, la cual representaremos por $\dot{\theta}$, trataremos de obtener la velocidad tangencial del vehículo durante el giro.

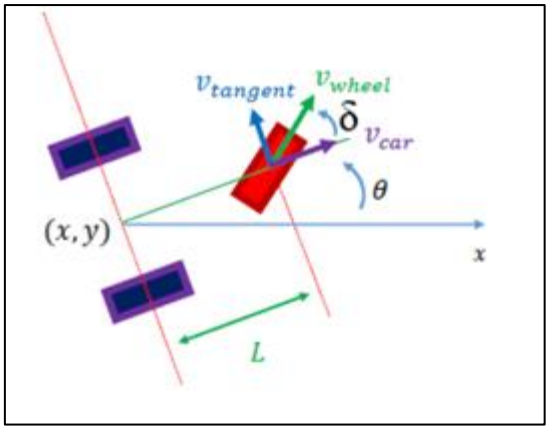


Ilustración 21: Descomposición velocidades Ackermann

En la imagen anterior vemos como se divide la velocidad en 3 componentes que ya se habían mencionado anteriormente: velocidad lineal del vehículo v_{car} (v), velocidad de la rueda frontal del vehículo v_{wheel}(u) y la velocidad tangente del vehículo al girar $v_{tangent}$.

De la imagen anterior podemos obtener la relación existente entre dichas velocidades:

- $u \cdot \cos(\delta) = v$
- $u \cdot \sin(\delta) = v_{tangent}$

Operando podremos representar la velocidad tangente en función de la velocidad lineal del vehículo, la cual podremos medir:

$$v_{tangent} = u \cdot \tan(\delta)$$

Teniendo en consideración que la velocidad tangencial de un cuerpo rotando con una velocidad ω y un radio de giro R, viene dado por $v_{tangent} = \omega \cdot R$, podremos obtener la velocidad de giro del vehículo $\dot{\theta}$:

$$\dot{\theta} = u \cdot \frac{1}{L} \cdot \tan(\delta)$$

Con esto quedaría definida la cinemática completa del vehículo de Ackermann.

2.4 Geometría Ackermann

En este apartado se pretende describir la geometría de Ackermann así como justificar su necesidad a la hora de diseñar un mecanismo de dirección. Seguidamente se describirán las características constructivas que definen un mecanismo que cumpla con la geometría de Ackermann.

Como ya se mencionó anteriormente, el vehículo de Ackermann se compone de 4 ruedas, montándose las 2 ruedas traseras sobre un mismo eje, mientras que las 2 ruedas delanteras actúan como ruedas directrices que pueden girar sobre un eje perpendicular al suelo. Para girar el vehículo, se giran las ruedas de forma que generen una curva con un radio 'R'.

Un problema que surge en esta situación es el hecho de que si ambas ruedas directrices giran el mismo ángulo, generarán radios de giro distinto, como podemos ver en la imagen inferior:

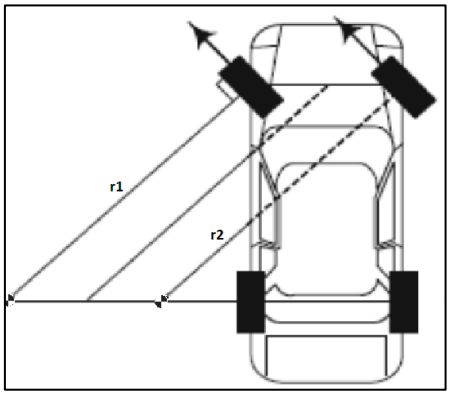


Ilustración 22: Mecanismo de dirección paralela.

Esto es lo que se conoce como sistema de dirección en paralelo.

Al estar girando con un radio de giro distinto para cada rueda, se provocará un deslizamiento de las ruedas que a altas velocidades podría suponer un grave peligro tanto para los pasajeros del vehículo como para cualquier otro vehículo cercano.

Para compensar este efecto, se diseña el mecanismo de giro de forma que el centro de giro de ambas ruedas se encuentre en el mismo punto, entrando aquí el mecanismo de dirección de Ackermann. En la siguiente imagen puede apreciarse el funcionamiento básico de este sistema:

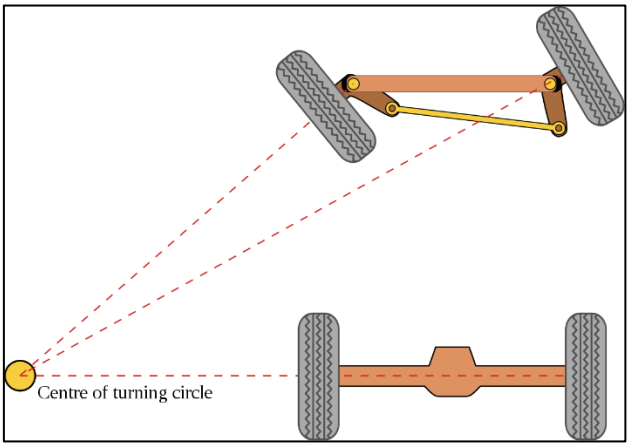


Ilustración 23: Mecanismo giro Ackermann

Como vemos se trata de un mecanismo de 4 barras unido por juntas rotativas, de modo que las 4 barras forman un trapecio. Al diseñar este mecanismo adecuadamente, se logra un sistema en el que ambas ruedas puedan girar formando el mismo radio de giro.

2.4.1 Diseño mecanismo Ackermann

Para dimensionar el mecanismo de dirección del vehículo nos basaremos en el artículo *Design of an Ackermann-type steering Mechanism* [1].

En la siguiente imagen podemos ver como funciona el mecanismo de Ackermann. Generando para cada rueda directriz, un ángulo con respecto de la vertical diferente. De forma que el centro de giro se encuentre en el mismo punto para ambas ruedas.

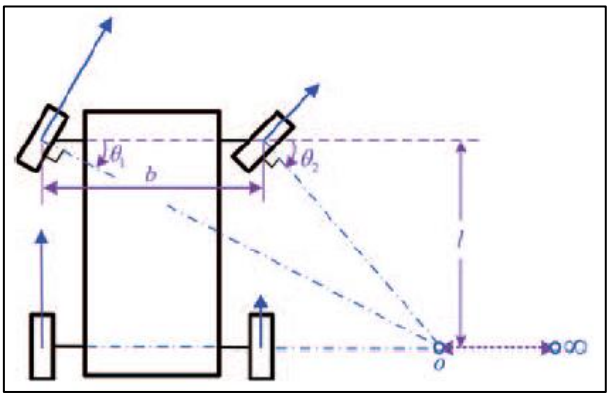


Ilustración 24: Geometría de Ackermann 1

Se aprecian además 2 de las primeras dimensiones que requeriremos definir del vehículo: el ancho de vías (*b*) y la distancia entre ejes (*l*). En el capítulo 3 se definen los valores de estas variables para el vehículo simulado.

En la siguiente imagen puede apreciarse la numeración que se usará en esta memoria para referirse a las distintas barras que componen el mecanismo de Ackermann:

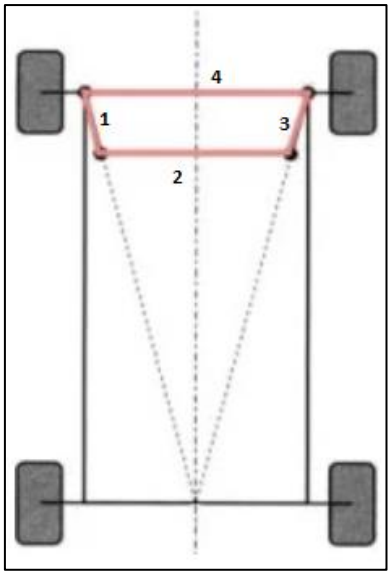


Ilustración 25: Geometría de Ackermann 2

Podemos ver que se trata de un mecanismo de 4 barras con 3 articulaciones de rotación. En este mecanismo, las ruedas directrices estarían acopladas a las barras 1 y 3, las cuales están unidas mediante la barra móvil 2 y la barra fija 4.

Si se diseña adecuadamente, las prolongaciones de las barras 1 y 3 deberían de cortarse en el centro del eje trasero del vehículo. Para lograr esto definiremos las barras 1 y 3 con una longitud y ángulos adecuados:

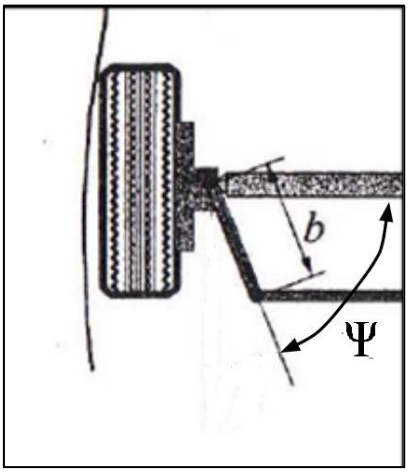


Ilustración 26: Diseño Ackermann

A la distancia *b* de la imagen la llamaremos *l1* y el ángulo ψ lo mediremos como el ángulo entre la barra 1 y la barra 4, al estar la rueda completamente perpendicular a la barra 4.

El único parámetro que faltaría por definir sería la longitud de la barra *l2*. Para obtener dicha longitud tomaremos en consideración las relaciones trigonométricas que se producen cuando el vehículo trata de girar por una curva:

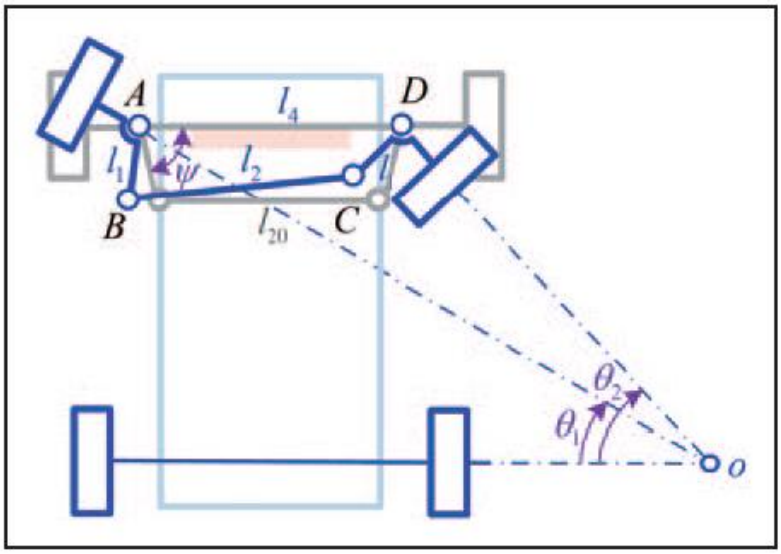


Ilustración 27: Geometría de Ackermann 3

En esta imagen se observan las limitaciones producidas así como los ángulos a considerar durante el giro utilizando un mecanismo de Ackermann.

En esta se puede observar que, al realizar un giro, cada rueda directriz girara un ángulo θ distinto. De **[1]** se obtiene que la longitud de la barra 2 vendrá dado por:

$$l_2 = l_4\sqrt{1 + 2\xi^2 + 2\xi^2\cos[2\psi + \theta_1 - \operatorname{arc\,cot}(\cot\theta_1 - \varsigma)] - 2\xi\{\cos(\psi + \theta_1) + \cos[\psi - \operatorname{arc\,cot}(\cot\theta_1 - \varsigma)]\}}$$

En donde:

- $\xi = \frac{l1}{l4}$
- $\varsigma = \frac{b}{l}$

De la función anterior se deduce que el valor de *l2* es proporcional al ángulo girado θ_1, es decir, que idealmente la longitud de la barra *l2* debería de variar conforme vaya girando la rueda. Ya que esto es físicamente imposible, definiremos θ_1 como 30º ya que es el mayor ángulo de giro que queremos que pueda realizar el vehículo.

2.5 Control automático

En el siguiente apartado describiremos las características básicas que componen un sistema de control PID, muy habitual es este tipo de proyectos. El controlador automático que se pretende diseñar para el proyecto, tratará de controlar la velocidad de un motor de 12 V, es por ello que orientaremos la explicación del controlador en torno a dicho proceso.

El control automático se compone de una fase en prácticamente cualquier proyecto de ingeniería en el que sea necesario controlar alguna variable de un proceso de forma no manual. Ejemplos de esto son el control de temperatura de una fundición en la industria siderúrgica, o la velocidad del motor de un avión.

Los sistemas de control se pueden realizar en lazo abierto o cerrado, siendo la distinción entre ellos la existencia de un sistema de realimentación de la variable a controlar en el lazo cerrado. Esto le da al sistema de control de lazo cerrado una mayor precisión frente al sistema de control en lazo abierto. Debido a que el proyecto incluirá un sistema de control en lazo cerrado, centraremos el siguiente apartado en torno a dicho tipo de controlador.

El aspecto general de un sistema de control cerrado es similar al de la imagen inferior:

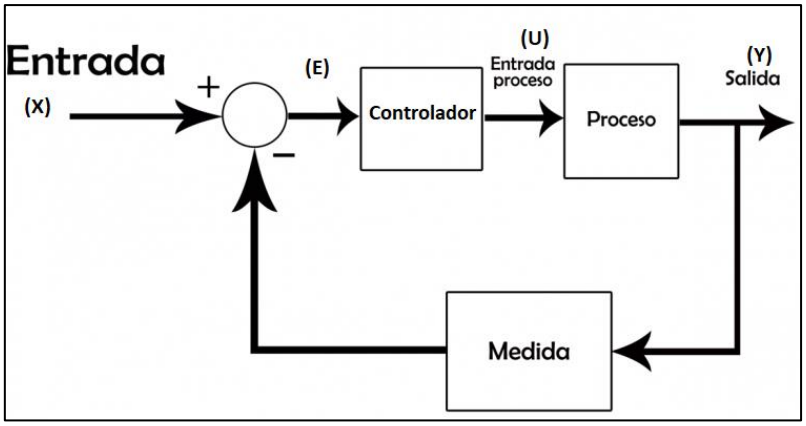


Ilustración 28: Control automático

En este sistema de control se pretende controlar una salida (Y) que para el caso que nos conviene será la velocidad del eje de salida de un motor eléctrico. El valor de salida la obtendremos mediante una Medida (sensor) valor el cual restaremos a la entrada (X), que indicará la velocidad deseada para el motor.

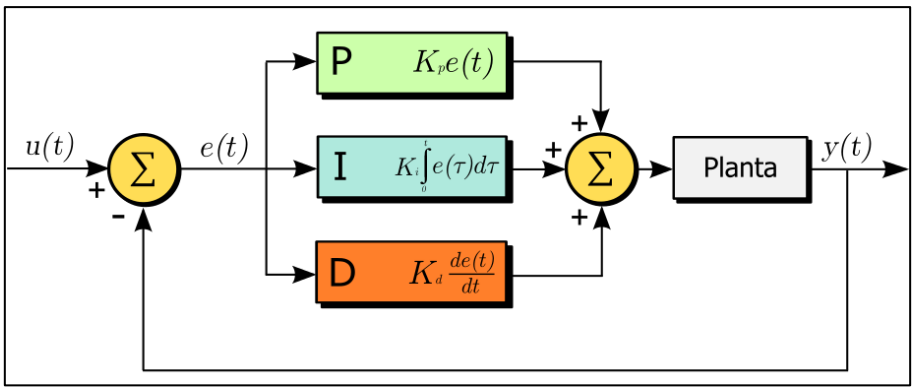
De esta resta se obtiene el error (E) producido en el sistema, para el caso que nos conviene este valor representa la diferencia entre la velocidad real del motor y la velocidad deseada del mismo.

Una vez obtenido el error, usaremos dicho valor para obtener la entrada(U) que se le requiere aplicar al proceso (en nuestro caso, el voltaje que se requiere aplicar al motor eléctrico) para compensar dicho error.

El controlador se implementa generalmente mediante un microcontrolador o directamente un controlador PID industrial, según el ámbito y proceso a controlar. El tipo de controlador mas habitual en la industria y en el campo de la robótica es el controlador PID.

2.5.1 PID

El controlador Proporcional-Integral-Derivativo es un algoritmo de control que trata de compensar el error que se produce en cualquier proceso realimentado.



Este tipo de algoritmo se compone de la suma de tres parámetros distintos, cada uno de ellos aportando un efecto distinto al sistema de control. Conviene mencionar que según el proceso a controlar, puede no ser necesario incluir las 3 partes del controlador, quedando con ello un controlador PI, PD o incluso un controlador P, simplemente con la acción proporcional.

Acción Proporcional

El valor de la parte Proporcional del sistema de control se obtiene al simplemente calcular el producto de la señal obtenida de error (E) por una variable K_P. Esta acción será entonces directamente proporcional al valor actual del error.

$$U_P = K_P \cdot E$$

Mediante este tipo de acción de control se logra una mejora general en el comportamiento transitorio y permanente del sistema de control, pero no se logra eliminar por completo el error producido en el proceso a controlar.

Acción Integral

Esta parte del PID, calcula el error que se ha producido anteriormente, de forma que el error acumulado en periodos anteriores aumentaría el valor de dicha acción de control. Para realizar esto, calcula la integral de la función del error E(t) hasta el tiempo actual. La fórmula para obtener dicha acción de control es la siguiente:

$$U_I = K_I \cdot \int_0^{t*} E(t)dt$$

En la siguiente imagen se puede apreciar de forma gráfica el funcionamiento de dicha acción de control:

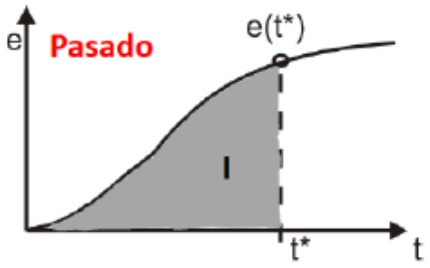


Ilustración 29: Acción Integral

Esta acción de control es la que le permite al controlador obtener un error nulo en régimen estacionario.

Acción Derivada

La acción de control derivada es la que trata de predecir el error que se va a producir en el siguiente período de tiempo. Para ello el algoritmo obtiene la derivada de la función del error en el tiempo actual. La fórmula para obtener dicha acción de control es la siguiente:

$$U_D = K_D \cdot \frac{dE(t)}{dt}$$

Mediante esta acción de control logramos mejorar el comportamiento del sistema de control en la etapa transitoria del mismo.

2.6 Seguimiento trayectoria

El control automático de un vehículo de Ackermann es un tema de gran actualidad para el cual existen varias publicaciones científicas que estudian el funcionamiento y limitaciones de cada método. En los coches modernos estos sistemas de conducción autónoma suelen disponer de un conjunto de cámaras y sensores que permiten que el vehículo sea capaz de conducir conforme todas las normas de tráfico vigentes sin causar peligro alguno para el resto de conductores ni para los viandantes.

En este apartado se pretende describir el trabajo realizado para obtener un método de control que sea capaz de lograr que el vehículo diseñado siga con la mayor precisión posible una trayectoria predefinida. Las propias trayectorias a seguir se describirán en el Capítulo 3.

Con ello lo que se pretende es, primeramente diseñar una trayectoria que el vehículo deberá ser capaz de seguir teniendo en cuenta las limitaciones de giro del mismo. Seguidamente se le enviará al vehículo la trayectoria que debe seguir.

2.6.1 Pure Pursuit

El método de control Pure Pursuit es un método de control bastante habitual para vehículos modelizados con el modelo de la bicicleta que se describió anteriormente. En estos artículos [2] y [3] se detalla como implementar este método de control de trayectoria.

Este método trata de obtener el ángulo de giro que requeriría la rueda del vehículo para poder alcanzar un punto en el plano mediante un único giro en el que el coche avance hacia adelante.

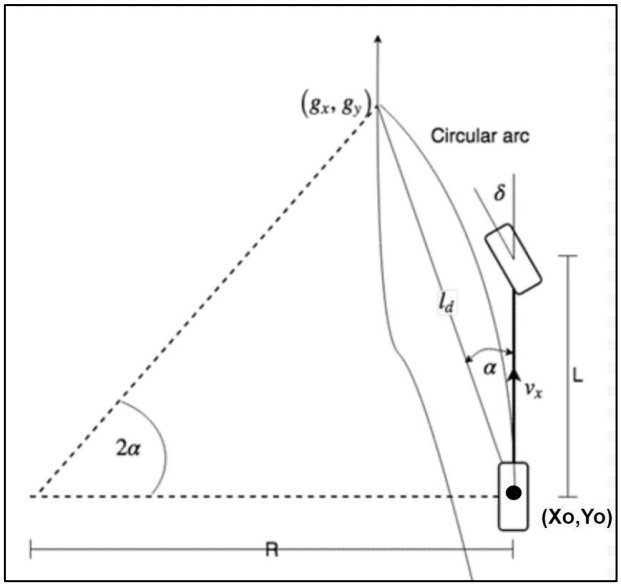


Ilustración 30: Pure Pursuit

En este método de control, el punto del vehículo que usaremos como coordenada a la hora de controlar el vehículo será el punto central del eje trasero.

El objetivo de este método es generar un radio de giro que pase por el punto objetivo así como por la coordenada característica, formando una tangente con la línea imaginaria que une los centros de los ejes traseros y delanteros. De esta forma se generará un arco de giro que una la coordenada objetivo con el punto central del eje trasero.

Llamaremos $\overrightarrow{l_d}$ al vector que une el punto objetivo con el punto central del eje trasero del vehículo.

De la ilustración anterior tenemos las siguientes variables:

- gx, gy: Coordenadas del punto por el que queremos que pase el vehículo.
- l_d : Módulo del vector $\overrightarrow{l_d}$.
- α: Ángulo que forma la línea que pasa por el centro del vehículo con el vector $\overrightarrow{l_d}$.
- δ: Ángulo de giro de la rueda delantera.
- L: Distancia entre ejes de nuestro vehículo.
- R: Radio que produciría el giro necesario para alcanzar el punto deseado.
- v_x: Velocidad lineal del vehículo

Recordemos que para obtener la distancia de $\overrightarrow{l_d}$ basta con calcular la distancia entre las 2 coordenadas que forman dicho vector:

$$l_d = \sqrt{(gx - Xo)^2 + (yx - Yo)^2}$$

Siendo (Xo, Yo) el punto central del eje trasero.

De la imagen anterior podremos calcular el Radio de giro que representa el arco por el que se debe desplazar el vehículo para pasar por cierto punto.

Aplicando el teorema del seno se obtiene:

$$\frac{l_d}{\sin(2\alpha)} = \frac{R}{\sin(\frac{\pi}{2} - \alpha)}$$

Teniendo en cuenta las siguientes identidades trigonométricas:

$$\sin(2\alpha) = 2 \cdot \sin(\alpha) \cdot \cos(\alpha) \qquad \sin\left(\frac{\pi}{2} - \alpha\right) = \cos(\alpha)$$

Tras simplificar se obtiene la siguiente fórmula para obtener el radio de giro, R, que debe producir el vehículo:

$$R = \frac{l_d}{2 \cdot \sin(\alpha)}$$

Del **apartado 2.3** se obtuvo que el ángulo de giro requerido para realizar un giro de radio R viene dado por la formula:

$$R = \frac{L}{\tan(\delta)}$$

Con lo que se concluye que el ángulo que deberán girar las ruedas para que el vehículo pueda realizar dicha maniobra de giro vendrá dado por:

$$\delta = \arctan\left(\frac{2 \cdot L \cdot \sin(\alpha)}{l_d}\right)$$

Mediante está formula podremos obtener un controlador que permita que el vehículo diseñado sea capaz de seguir una trayectoria que definiremos más adelante.

En el capítulo 3 se definirá con mayor detalle el modo de implementar dicha ecuación en Simulink.

3 Simulación

Este capítulo abarcará todos los temas relativos a la simulación del vehículo diseñado. Se comenzará el capítulo describiendo el vehículo diseñado para la simulación 3D así como el ensamblaje completo del mismo.

Seguidamente describiremos detalladamente como se ha transferido el modelo diseñado en SolidWorks al entorno de programación de Simulink Simscape. Se incluye además una descripción de como se ha implementado en Simulink el sistema de control de trayectoria que se introdujo en el capítulo 2.

Finalmente se muestran los resultados obtenidos para la simulación realizada del vehículo diseñado.

3.1 Modelo 3D Diseñado

A continuación, describiremos el vehículo diseñado para este proyecto. Conviene mencionar que el vehículo que veremos a continuación es el vehículo mediante el cual se han realizado las simulaciones y puede ser distinto del modelo real construido. A pesar de ello se tratará de que el modelo real tenga la mayor similitud posible con el modelo aquí mostrado.

Algunas consideraciones que se han debido de tomar en cuenta durante el diseño han sido las siguientes:

- Debido al tamaño de la impresora 3D que usaremos para crear la base del vehículo limitaremos el largo y el ancho de la misma para que tenga un tamaño similar a una hoja estándar A4.
- Para la simulación se han escogido unas ruedas de plástico con un diámetro de 65 mm muy habituales en este tipo de proyectos.
- De forma similar a un vehículo real, las ruedas tendrán un ángulo de giro máximo de 30º.
- Ya que el modelo diseñado en este apartado está diseñado para comprobar el funcionamiento del vehículo en la simulación, no se ha visto necesario escoger los tornillos que unen las piezas ni los eslabones. Esto se verá más adelante en la implementación real del vehículo.

Primero comentaremos brevemente la base diseñada para el vehículo, ya que es el componente que une las 2 partes en las que se separa el vehículo, seguidamente se comentarán por separado las partes de tracción y dirección del vehículo.

Base

La base del vehículo tiene la función de actuar a modo de 'esqueleto' del vehículo, de forma que sobre la misma se puedan montar el resto de componentes del vehículo.

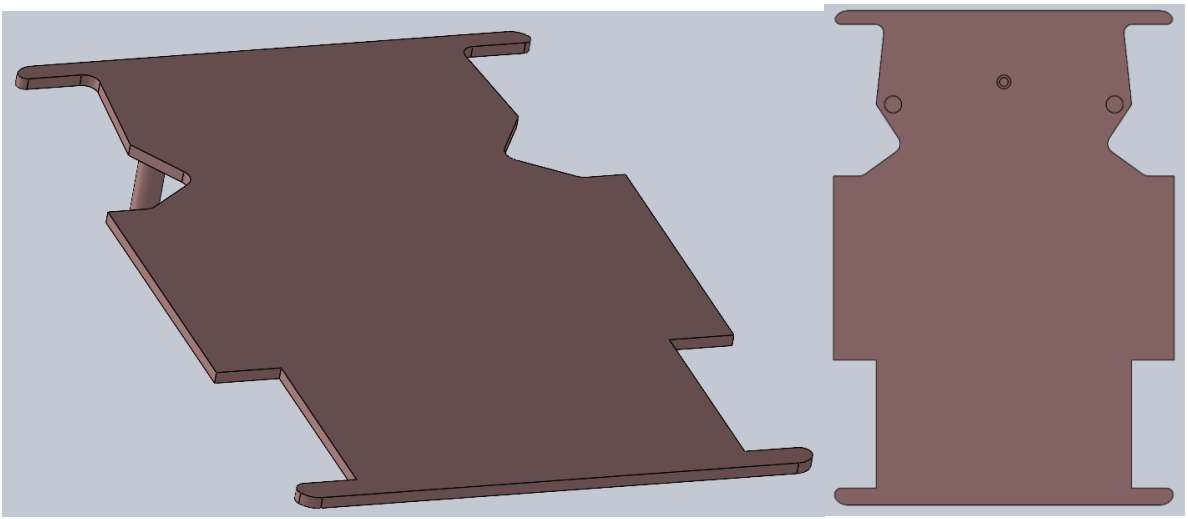


Ilustración 31: Base del vehículo

La base resulta en una única pieza simétrica sobre la que se montan los componentes del vehículo. En esta se toma en consideración el dejar espacio suficiente para el giro de las ruedas así como la correcta colocación de los componentes relevantes del sistema de dirección.

Las piezas aquí mostradas corresponden tanto a componentes que han sido creados mediante el programa SolidWorks por el estudiante, como piezas comerciales que han sido obtenidas de librerías de dibujo 3D tal y como GrabCAD. Se hará una distinción entre aquellas piezas que requieran ser impresas en 3D y las que por necesidades constructivas o económicas requieran ser componentes comerciales. Los planos de las piezas creadas con Impresión 3D se encuentran en el Anexo I.

3.1.1 Mecanismo de dirección

El mecanismo de dirección, como ya sabemos, es el conjunto de componentes del vehículo que se encarga de dirigir las ruedas del vehículo de forma que el vehículo se desplace en una dirección u otra.

En la siguiente imagen puede apreciarse la vista superior del mecanismo de dirección diseñado:

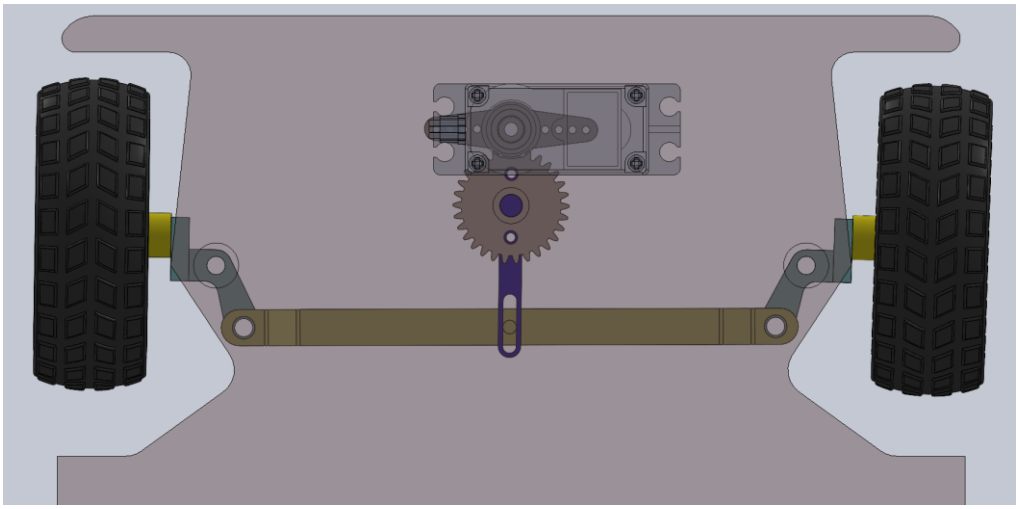


Ilustración 32: Mecanismo de dirección Imagen 1

Además del propio mecanismo de Ackermann, se ha requerido diseñar el mecanismo que permite controlar la dirección mediante un servomotor. Para ello se ha diseñado el mecanismo que se puede apreciar en la siguiente imagen:

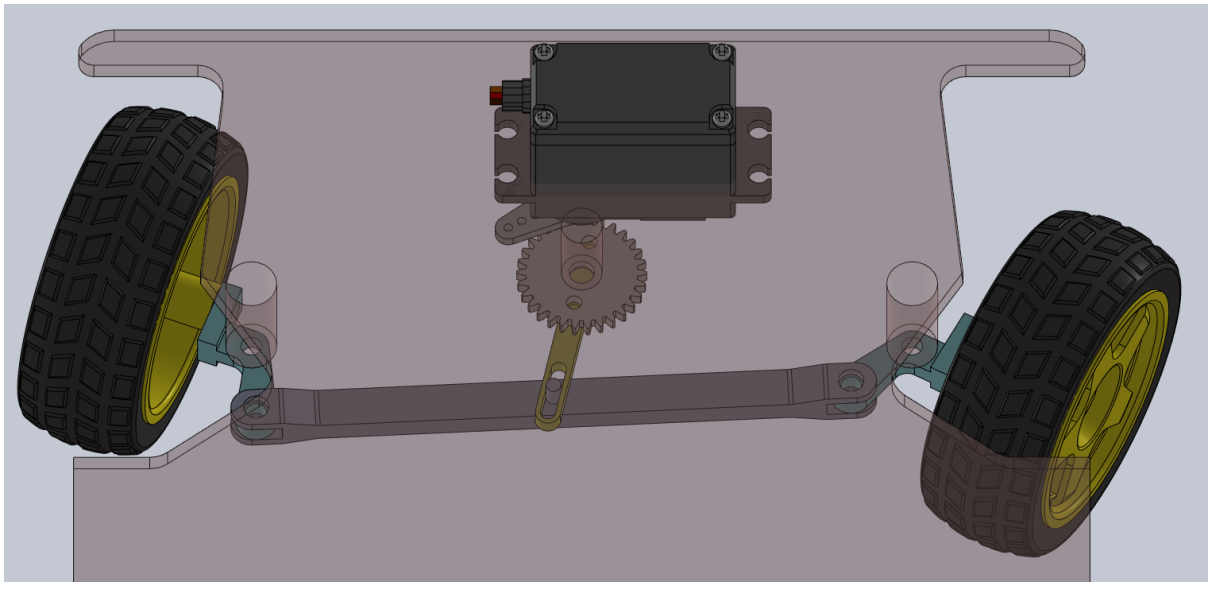


Ilustración 33: Vista superior sistema dirección

El mecanismo se trata de un juego de 2 engranajes, uno pequeño montado directamente sobre un servomotor y otro conectado a una barra que se monta sobre la barra 2 del mecanismo de Ackermann, formando una articulación de traslación, que provocará mediante el giro del servomotor, que las ruedas giren hacia un lado u otro.

El motivo de que se haya usado un par de engranes en vez de conectar la barra amarilla directamente al servomotor, es para permitir que el ángulo de giro que podremos aplicar al servomotor sea lo más amplio posible. Esto nos permitirá realizar unos giros más suaves y precisos.

A continuación, se muestra el proceso seguido para diseñar el mecanismo de dirección que cumpla con las características requeridas por la geometría de Ackermann.

En la siguiente tabla podemos recordar las variables más relevantes para definir geométricamente el mecanismo de dirección de Ackermann:

Descripción	Símbolo
Distancia entre el eje frontal del vehículo y el trasero	*l*
Distancia entre los puntos de apoyo de las ruedas	*b*
Longitud bieletas ruedas izquierda/derecha	*l1/l3*
Longitud barra que une las bieletas	*l2*
Distancia entre los puntos de giro de las bieletas	*l4*
Angulo que provocan las bieletas con el eje imaginario que une las ruedas directrices	ψ
Angulo de giro escogido para definir *l2*	θ_1

La definición de algunas de estas variables queda a elección del diseñador, mientras que el resto vendrán definido por ecuaciones que ya se introdujeron en el capítulo anterior. Recordando las consideraciones iniciales que se describieron al principio del capítulo, podemos definir las primeras variables de nuestro vehículo:

- *b*: 180 mm
- *l*: 195 mm
- *l1*: 15 mm
- *l4:* 130 mm

El resto de las variables se calcularán a partir de dichos valores:

- Ψ: 1,138 rad. Obtenido en base al ancho y largo del vehículo mediante la siguiente función:
  - $\Psi = \tan^{-1}\left(\frac{2 \cdot l}{b}\right)$

  Teniendo en consideración la fórmula:

$$l_2 = l_4\sqrt{1 + 2\xi^2 + 2\xi^2\cos[2\psi + \theta_1 - \operatorname{arc\,cot}(\cot\theta_1 - \varsigma)] - 2\xi\{\cos(\psi + \theta_1) + \cos[\psi - \operatorname{arc\,cot}(\cot\theta_1 - \varsigma)]\}}$$

  Y las constantes:

- $\xi = \frac{l1}{l4} = 0.1153$
- $\varsigma = \frac{b}{l} = 0.9252$

- θ_1 = 30º
  Se obtiene la longitud de la barra 2:
- *l2*: 117.22 mm

A partir de estos datos podremos diseñar la base del propio vehículo, así como el trapezoide que compone el mecanismo de dirección de Ackermann.

En la imagen inferior puede apreciarse como quedaría el mecanismo de Ackermann con las longitudes obtenidas anteriormente.

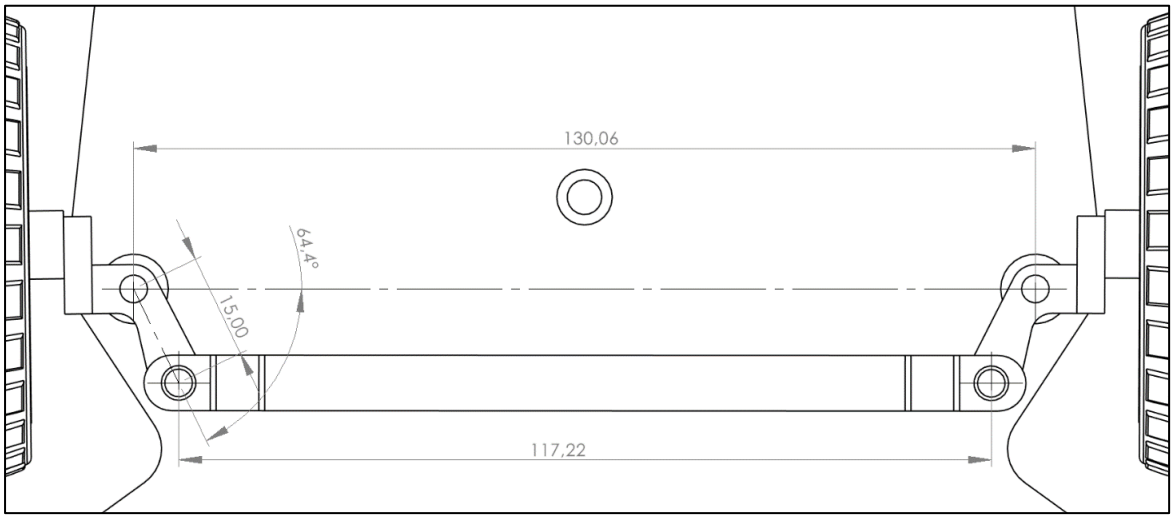


Ilustración 34: Mecanismo Ackermann

A continuación, veremos el modelo 3D que se ha realizado para cada componente del sistema de dirección.

Barras 1 y 3 (bieletas)

Estas 2 barras están formadas por la misma pieza, la cual puede apreciarse en la imagen inferior:

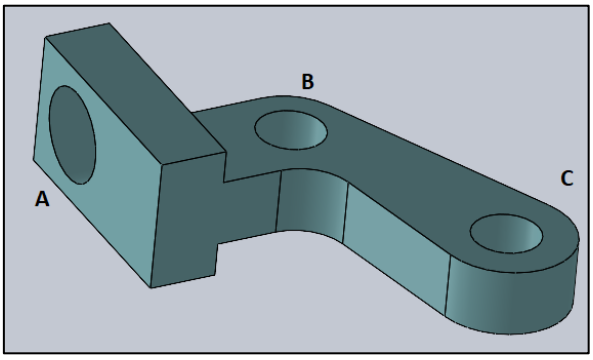


Ilustración 35:Modelo barras 1 y 3

Como podemos apreciar, la pieza está formada por 3 agujeros. Un agujero A en el que se monta la rueda, un agujero B que forma un par de rotación con la base del vehículo y un agujero C que se monta mediante un par de revolución sobre la barra 2. Como ya se mencionó anteriormente, esta pieza requería de unas dimensiones concretas entre los agujeros B y C:

- Longitud l1: 15 mm
- Ángulo Ψ: 1,138 rad

Estos valores podrían traducirse a una distancia lineal en el eje X y otra distancia lineal en el eje Y:

- L1x = 15mm · cos(1,138) = 6.29 mm
- L1y = 15mm · sin(1,138) = 13.61 mm

Estas dimensiones pueden apreciarse en la siguiente imagen:

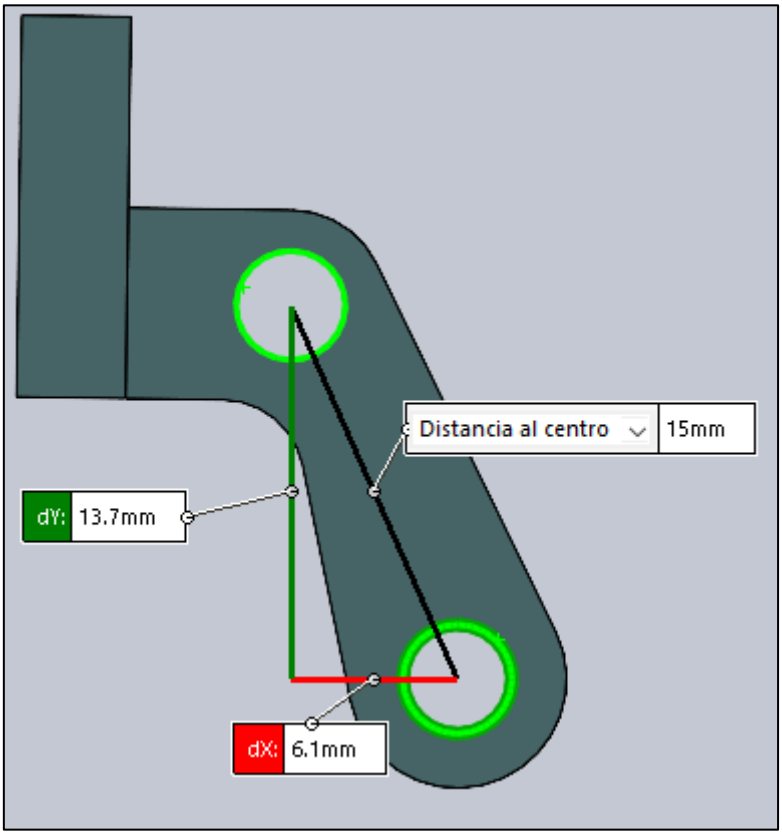


Ilustración 36: Dimensiones L1x y L1y

Debido a las características dimensionales de esta pieza, se obtendrá la misma mediante impresión 3D.

Barra 2

Esta barra es la que une las barras 1 y 3 descritas anteriormente, asegurando que siempre giren simultáneamente. Además de unir las barras 1 y 3, dispone de un saliente en la parte central que se conecta a la barra de dirección del conjunto del servo.

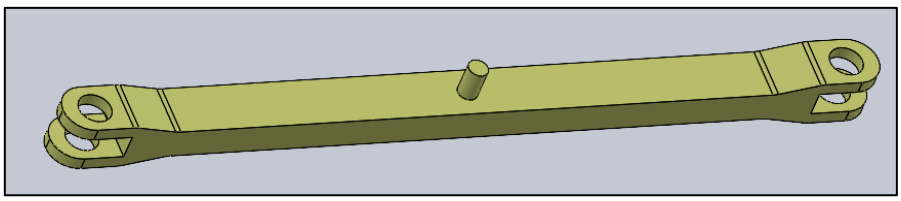


Ilustración 37: Barra 2

Este saliente permite generar un par de rotación-traslación que transfiere el movimiento rotatorio del Servomotor al giro de las ruedas del vehículo.

La longitud característica de este componente es la longitud entre los centros de los agujeros de los extremos de la barra, que como se comentó anteriormente deben estar a 127,22 mm.

Debido a las características dimensionales de esta pieza, se obtendrá la misma mediante impresión 3D.

Ruedas

Las ruedas que se han escogido para realizar el modelo de simulación son unas ruedas de plástico de 65 mm de diámetro y un ancho de 25 mm.

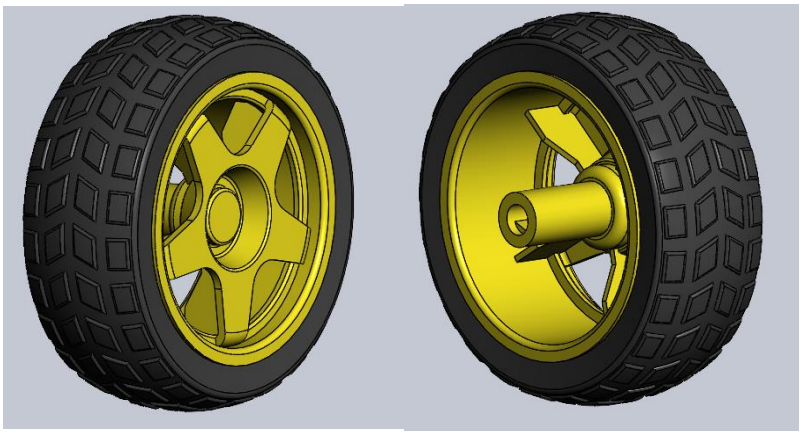


Ilustración 38: Ruedas

Estas corresponden a un modelo comercial de ruedas muy común para proyectos de robótica sencillos. El modelo de CAD se ha obtenido de una librería de piezas online.

Brazo dirección

Esta barra es la que genera junto a la barra 2 el par de rotación traslación mediante el que se transmite el movimiento giratorio del servomotor a la barra 2 que se encuentra en el mecanismo de Ackermann.

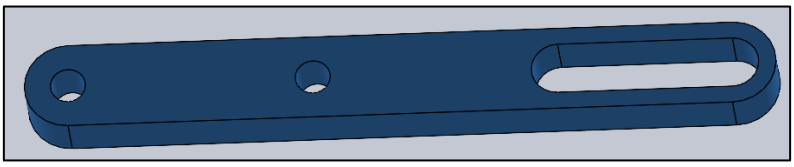
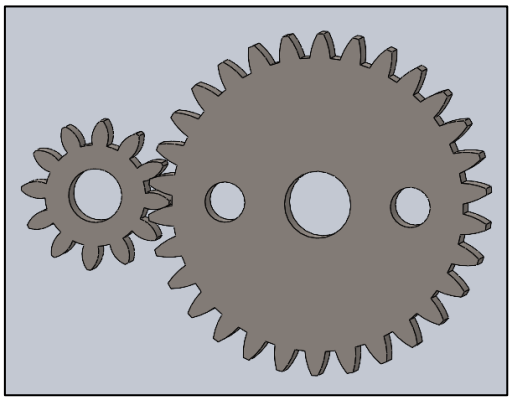


Ilustración 39: Brazo dirección

Esta barra se conecta mediante sus 2 agujeros al engranaje mayor del conjunto de engranes que describiremos a continuación y el agujero alargado se monta sobre la extensión que mencionamos anteriormente de la barra 2.

Conjunto de engranajes

Este conjunto de engranes conectan el servomotor con la barra de dirección anterior.



El engranaje pequeño se monta directamente sobre el servomotor mientras que el engranaje grande se monta sobre la base del vehículo. Dado que este componente será comercial, es muy probable que el conjunto aquí mostrado, el cual tiene una relación de transmisión de 2:5, no coincida exactamente con el modelo real creado.

El modelo 3D de ambos engranes se han obtenido mediante la herramienta Toolbox de SolidWorks.

Servomotor FS5109M

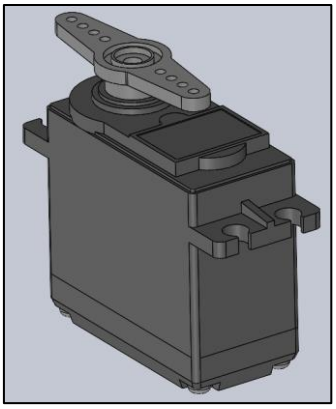


Ilustración 40: Servomotor

Este servomotor irá montado sobre la base del vehículo y será el componente que se encargue de dirigir el sistema de dirección del vehículo. El modelo del servomotor es probable que no coincida exactamente con el modelo real.

El modelo de CAD se ha obtenido de una librería de piezas online.

3.1.2 Mecanismo de Tracción

Además del propio sistema de dirección, el vehículo requiere de un sistema de tracción, para lo cual montaremos un motor eléctrico en la parte trasera del vehículo. En la imagen inferior se puede observar el sistema de tracción diseñado.

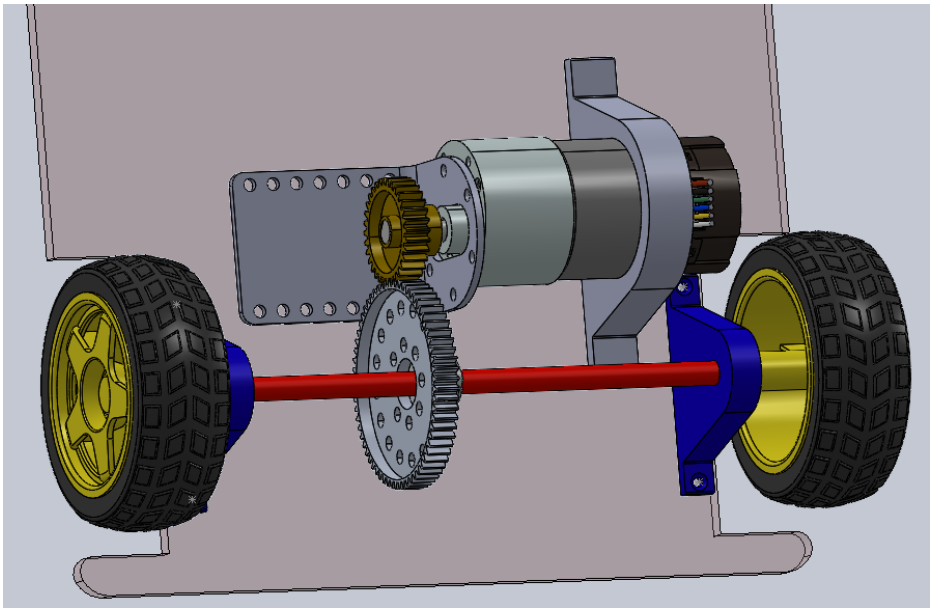


Ilustración 41: Sistema de tracción

A continuación, veremos los componentes que se han escogido o diseñado para realizar la simulación del vehículo.

Motor

El motor escogido para la simulación corresponde a un motor de la marca Pololu de 12 V con una velocidad máxima de 67 rpm. Este motor será lo suficientemente potente como para impulsar las ruedas del vehículo sin elevar excesivamente el presupuesto final del conjunto.

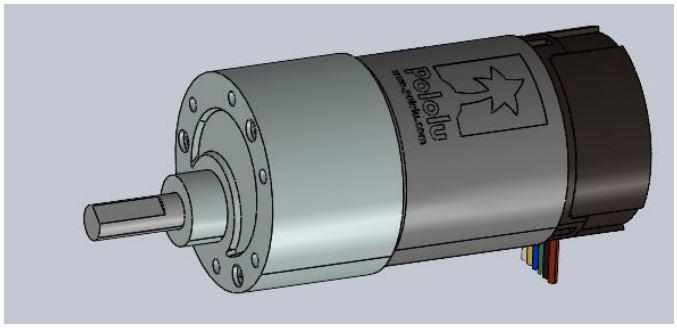


Ilustración 42: Motor vehículo

A nivel de simulación este componente no tendrá gran relevancia ya que podríamos aplicar el par de giro directamente sobre el eje del motor, por ello no se incluye en el montaje realizado con Simulink. Sin embargo se ha decidido incluirlo en este apartado para comprobar el espacio que ocupa, para ser considerado antes del montaje. El eje del motor tiene un diámetro de 6 mm y un hueco en forma de D, que permite transmitir el par al componente que se le acople.

Ruedas

Las ruedas del mecanismo de tracción serán las mismas que para el mecanismo de dirección, es decir unas ruedas de Ø65mmx25mm.

Piñón

El piñón es el engranaje de menor tamaño, que acoplaremos directamente al eje de salida del motor. Se ha escogido un modelo de engranajes de la marca Actobotics:

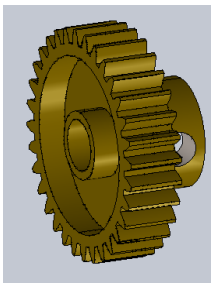


Ilustración 43: Piñon

Este engrane dispone de 32 dientes y un diámetro interno de 6 mm, con lo que posteriormente podremos montarlo sin problema alguno en el montaje real que se vaya a realizar.

Rueda

La rueda del conjunto de engranes será la que se monte sobre el eje de las ruedas traseras. Para ello se ha escogido un engranaje, también de la marca Actobotics, para facilitar el montaje entre los 2 engranes:

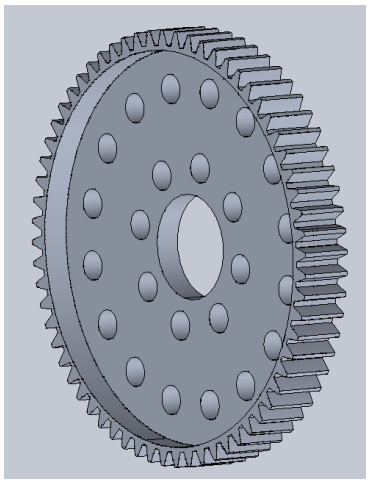


Ilustración 44: Rueda

Dicho engranaje dispone de un total de 64 dientes, con lo que la relación de transmisión del conjunto de engranes sería de 1:2.

Buje de tornillo

Para montar el engrane sobre el eje del motor en el vehículo real se usará un buje de tornillo que se puede apreciar en la siguiente imagen:

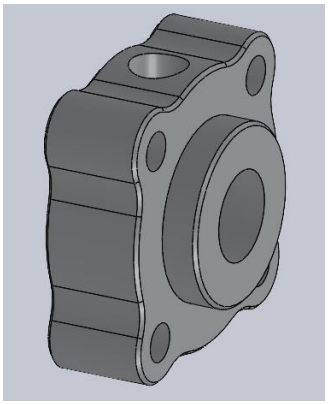


Ilustración 45: Buje de tornillo

Este componente no se incluirá en la simulación ya que su función no se considera indispensable para la misma e incluirlo podría incrementar el tiempo de computación.

Soportes del Eje

Estos soportes se crearán mediante la impresora 3D y serán los responsables de sujetar la barra del eje a la base del vehículo.

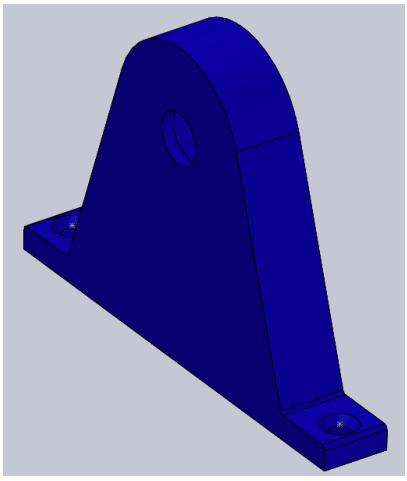
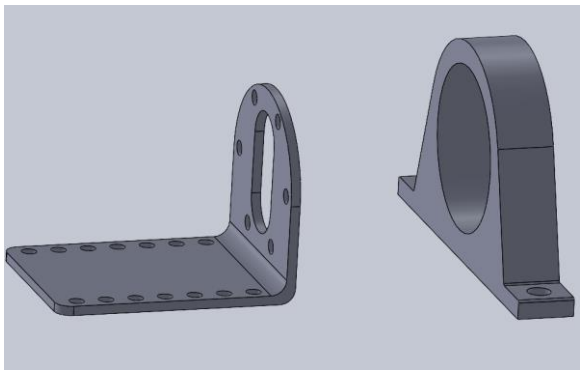


Ilustración 46: Soporte eje

En su parte inferior incluyen unos agujeros gracias a los cuales podremos montar los soportes a la base del vehículo mediante unos tornillos.

Soportes motor

A pesar de no aportar nada a la simulación, pero debido al mínimo grado de realismo que requiere el diseño, se ha decidido incluir en el mismo 2 soportes de motor que se utilizarán en el montaje real del vehículo:



El soporte de la izquierda es un soporte especifico de Pololu para el motor escogido y actuará como soporte principal. Se ha diseñado además un soporte adicional que tratará de reducir la oscilación del motor en el montaje final. El segundo soporte se creará mediante la impresora 3D. Ambos soportes disponen de agujeros para ser montados sobre la base mediante tonillos.

3.1.3 Ensamblaje completo

El ensamblaje completo que se ha diseñado puede apreciarse en las siguientes imágenes:

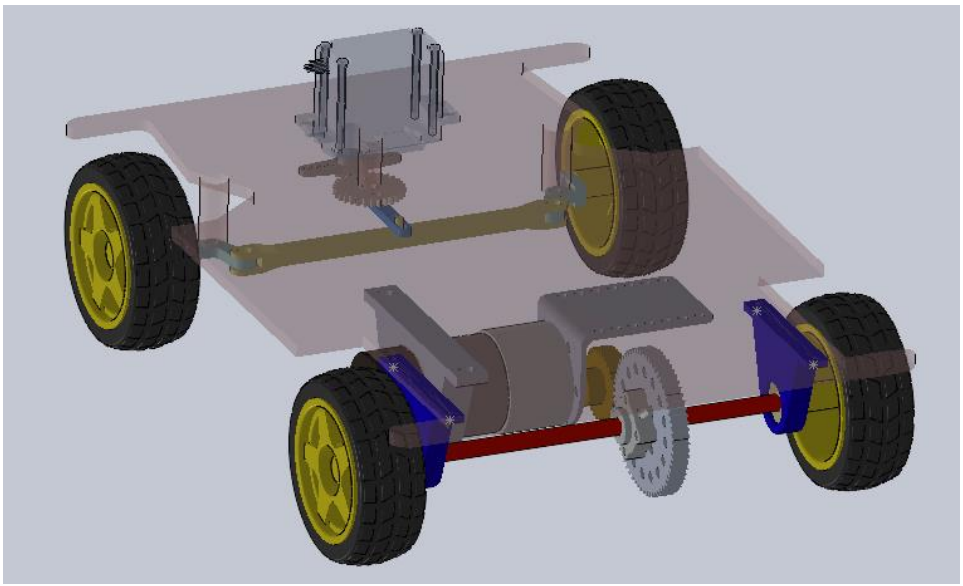


Ilustración 47: Ensamblaje Completo – vista ISO

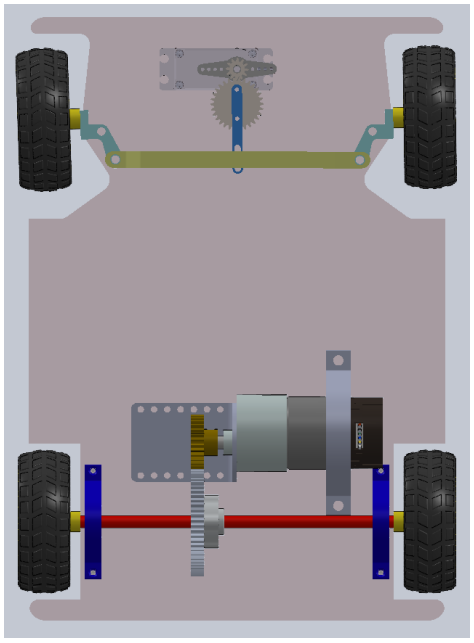


Ilustración 48: Ensamblaje Completo -Inferior

Cabe mencionar que este diseño resulta en un primer prototipo del vehículo que se usará en la simulación y que pueden aparecer cambios en el montaje final del vehículo.

3.2 Modelo Simscape Multibody

Para realizar la simulación del modelo diseñado anteriormente usaremos el programa MatLab. Dentro de dicho programa usaremos el complemento Simulink junto a la librería Simscape Multibody para simular el comportamiento del vehículo completo.

MATLAB es un entorno de programación utilizado extensamente en el ámbito de la ingeniería debido a su gran variedad de herramientas para diseñar modelos de simulación, así como controladores para diversos procesos. Dentro del ámbito de MATLAB se encuentra el entorno Simulink, que trata de un entorno de programación visual. Este será el entorno que más aprovecharemos en el trabajo.

A continuación, veremos el funcionamiento básico de Simulink, así como el complemento Simscape, que nos permitirá simular los componentes 3D.

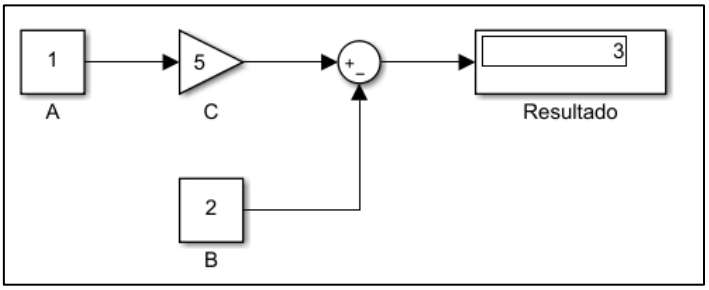


Ilustración 49:Modelo básico Simulink

En la imagen anterior puede apreciarse un diagrama sencillo creado con Simulink, mediante el que realizamos la siguiente operación aritmética:

$$A \cdot C - B = Resultado$$

El programa permite mediante la utilización de bloques, la generación de funciones que el ordenador resuelve numéricamente. Estos bloques pueden representar operaciones matemáticas simples (suma, resta,…) o pueden incluir operaciones mas complejas como veremos mas adelante (operaciones trigonométricas, integrales, derivativas,…).

El complemento Simscape nos permite hacer lo mismo pero en vez de usar bloques que representan operaciones aritméticas, utilizando componentes reales tales como motores eléctricos, tuberías, resistencias eléctricas, ruedas, etc. Todas ellas se encuentran modelizadas en el programa. En la imagen inferior se aprecia la modelización de un motor de corriente continua junto a su fuente de tensión y sensores:

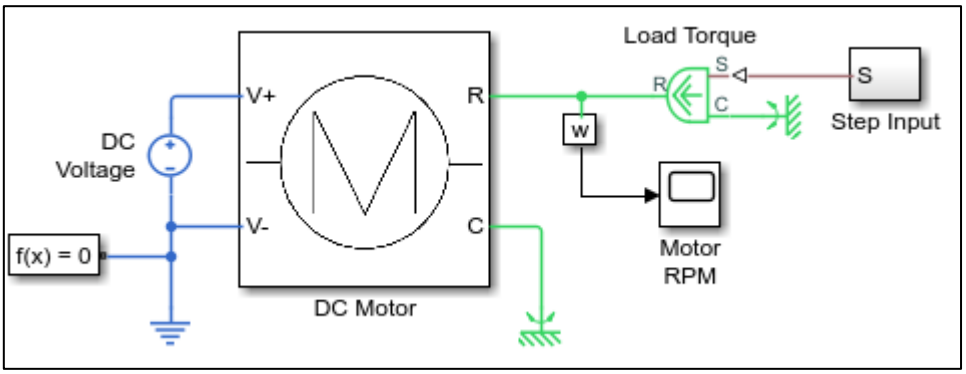


Ilustración 50: Simscape

A continuación, describiremos el modelo de vehículo que se ha creado en Simulink para comprobar el funcionamiento del vehículo en un entorno similar al entorno real en el que funcionará el vehículo que se decida montar.

3.2.1 General

Como ya se mencionó anteriormente, el vehículo se ha montado en el entorno de programación de Simulink, utilizando bloques de la librería Simscape, la cual permite realizar simulaciones con componentes físicos que interactúan entre ellos. A estos bloques les uniremos bloques de la librería básica de Simulink que nos permitirá diseñar además los sistemas de control que requiere el vehículo.

El objetivo principal de estas simulaciones es la de comprobar el adecuado funcionamiento del sistema de conducción autónoma diseñado, así como la posibilidad de comprobar las limitaciones que podría tener nuestro vehículo en un entorno real.

Cabe mencionar que algunos de los componentes descritos en el apartado anterior se han visto modificados o directamente eliminados con el fin de reducir el tiempo de simulación del vehículo.

Para simplificar el modelo y poder mejorar la comprensión del mismo, se han realizado varias subdivisiones que separan las partes funcionales del modelo en distintos subsistemas. En el nivel superior de la jerarquía se encuentran visibles todas las partes que componen nuestro modelo:

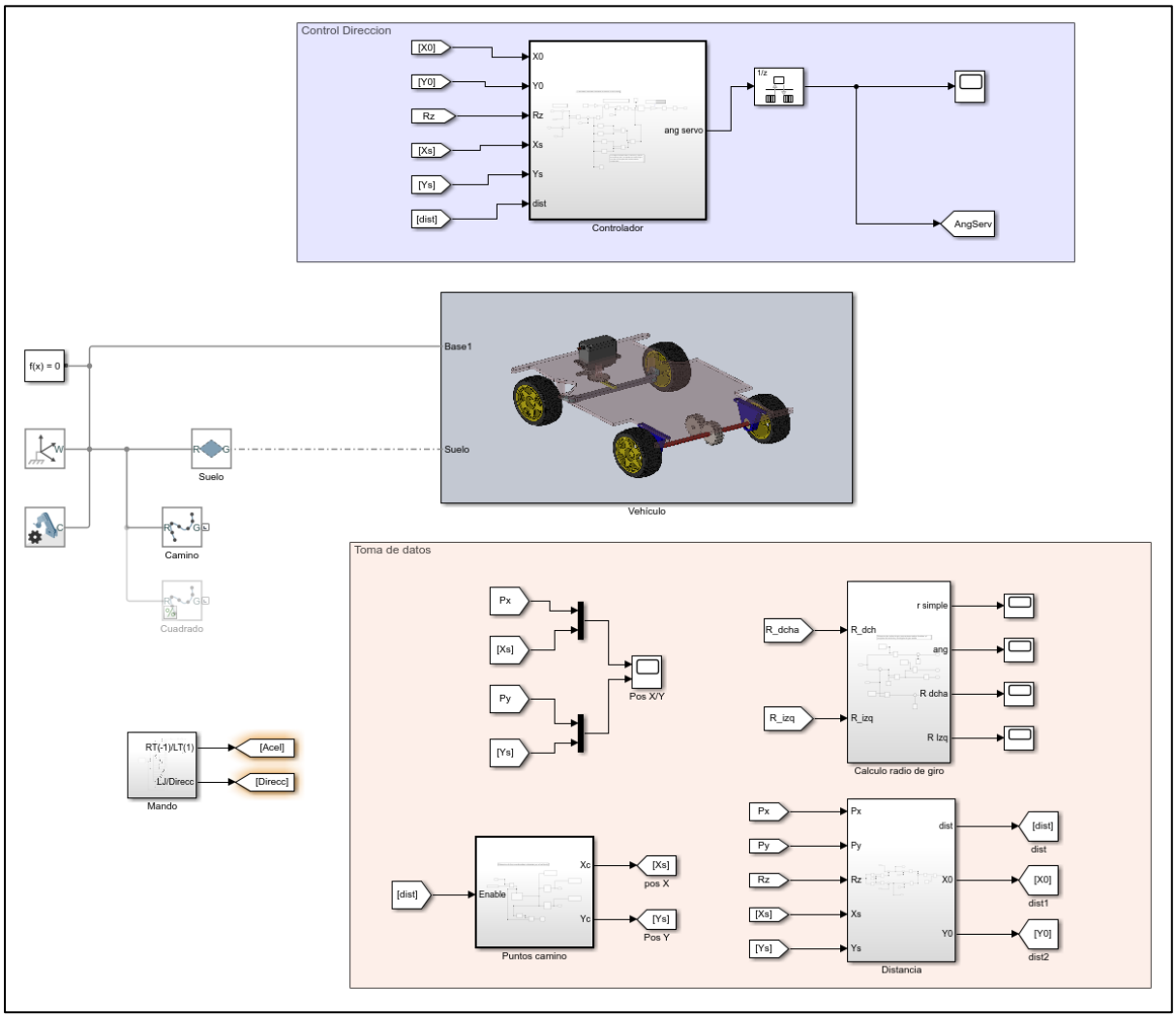


Ilustración 51: Modelo Simscape - Nivel superior

En esta imagen se pueden apreciar las principales partes que componen nuestro sistema:

- Bloques generales
- Sistema de control de dirección
- Vehículo simulado
- Toma y muestra de datos
- Control mediante mando

En el conjunto de bloques generales se incluyen los bloques que permiten la adecuada simulación de los bloques de Simscape. Además de ello se incluyen el bloque que representa el suelo sobre el que se desplazará el vehículo, así como un bloque que usaremos para representar visualmente la trayectoria que se pretende que siga el vehículo.

Para mayor comprensión del sistema, describiremos primero los 4 bloques de la librería de Simscape más comunes que nos encontraremos en el ensamblaje realizado, así como el concepto de 'frames' mediante el que se unirán los distintos componentes.

Solid

Este tipo de bloques se utiliza para representar cualquier componente que hayamos diseñado en el programa SolidWorks importándolo como un archivo en formato STEP. Para ello es necesario cargar en cada bloque Solid el modelo que representa el bloque que queremos modelizar. De esta forma podremos unir las distintas piezas del modelo y observar visualmente su comportamiento en la simulación

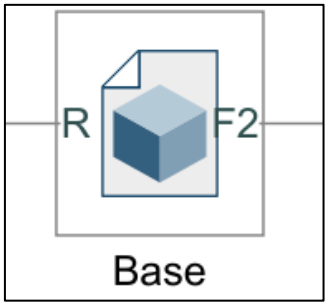


Ilustración 52: Bloque Solid

Los bloques Solid dispondrán siempre como mínimo de un puerto que usaremos para conectar el componente a otras partes del conjunto que se pretende simular. Estos puertos suelen representar partes concretas del solido 3D. como veremos a continuación.

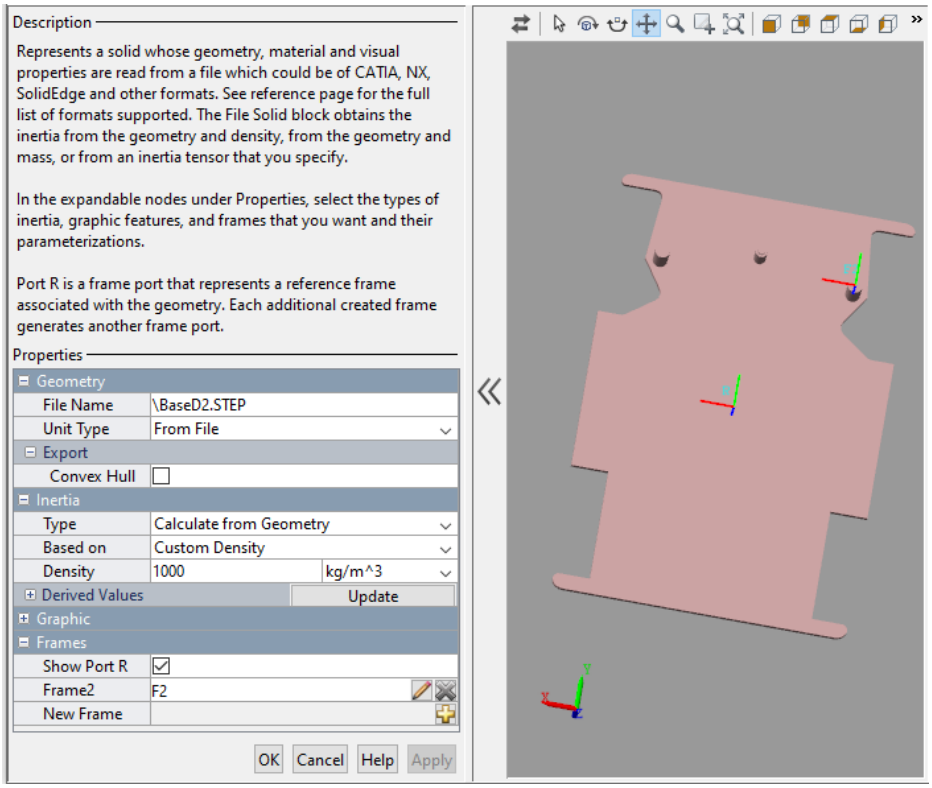


Ilustración 53: Parámetros bloque Solid

En esta imagen se muestran los parámetros más relevantes para la simulación que serán necesarios adaptar para cada componente que se requiera modelizar. Para cargar el modelo 3D en el bloque es necesario introducir la dirección del mismo en la primera línea de 'File Name'. Seguidamente podremos definir algunos aspectos mecánicos del modelo tales como la densidad del modelo y el modo de cálculo de la Inercia. También se nos permite exportar la geometría de la pieza, algo que nos servirá posteriormente a la hora de simular el contacto de las ruedas del vehículo con el suelo.

Seguidamente se muestran los distintos puntos de conexión que tendrá el modelo, que vendrá representado por 'Frames'. Estos puntos de conexión son los puntos en los que se conectarán el resto de componentes. Cualquier modelo cargado en un bloque Solid, tendrá siempre un puerto 'R' de forma predeterminada en el centro de masa del propio bloque. Se pueden definir Frames adicionales en puntos geométricos concretos del modelo tal y como el Frame F2, que se encuentra en el apoyo derecho de las ruedas.

Dichos frames representan, como puede verse en la imagen anterior, sistemas de coordenadas XYZ que nos facilitarán conectar otros modelos entre ellos así como diseñar adecuadamente los ejes de revolución del vehículo a simular. Dichos sistemas de coordenadas presentan un origen normalmente definido por alguna característica geométrica del componente sobre el que se encuentra tal y como el centro de la base cilíndrica que se observa que dispone la base.

El modelo aquí mostrado, representa la base del vehículo, sobre la que se conectarán los distintos componentes del vehículo

Rigid Transform

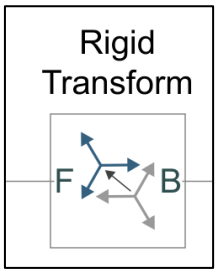


Ilustración 54: Bloque Rigid Transform

Este bloque tiene la función de representar una traslación o rotación de un punto en el espacio con respecto a otro. Se conecta un primer componente a la base (puerto B) y el frame del objeto que se quiera desplazar o girar se conecta mediante el puerto F al puerto adecuado del objeto deseado.

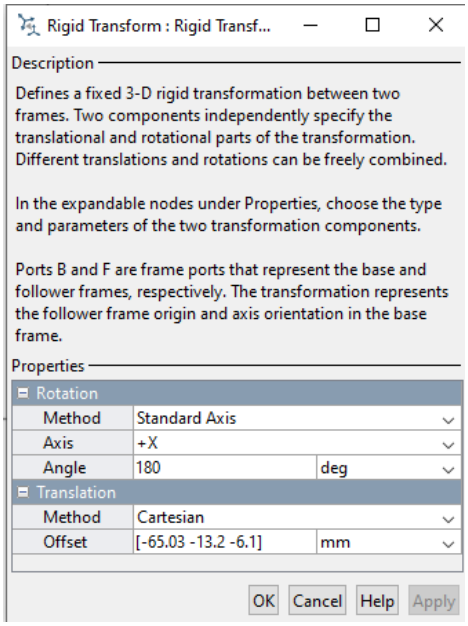


Ilustración 55: Parámetros Rigid Transform

Este bloque resulta de gran utilidad ya que permite Colocar piezas con una cierta distancia respecto de otra, ayudando con ello a montar el mecanismo que se diseñó en el apartado anterior. Cabe mencionar que los nodos conectados mediante un bloque de este tipo, no permiten ningún tipo de movimiento relativo entre ellos, cualquier movimiento aplicado a un nodo, se transmitirá al otro.

Articulación de revolución

Este bloque como su propio nombre indica, representa una articulación de revolución entre 2 piezas, permitiendo con ello el giro entre ellas.

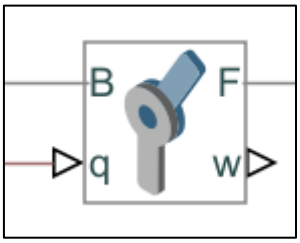


Ilustración 56: Bloque articulación de revolución

El bloque se compone de una puerto F (Frame) y un puerto B (Base), a los que se conectan puertos de otros componentes giratorios. El funcionamiento del bloque se basa en mantener los orígenes de ambos componentes conectados en el mismo punto y además obliga a que en todo momento, los ejes Z de los frames conectados sean coincidentes, pudiendo rotar ambos componentes alrededor de dicho eje conjunto.

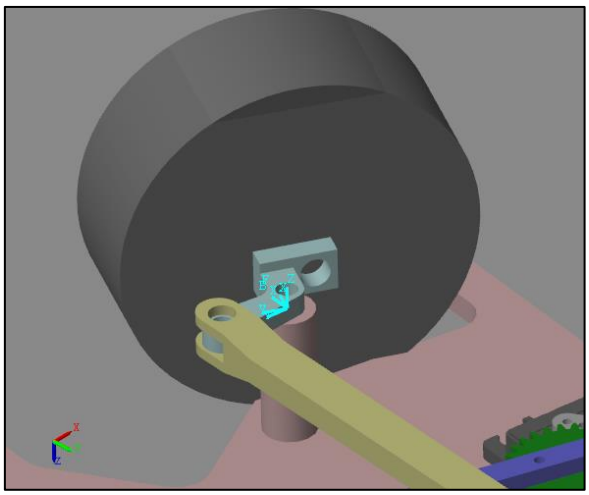


Ilustración 57: Juntas de revolución

En la imagen anterior se puede observar una junta de revolución creada mediante el bloque descrito. Dicho bloque permite que el mecanismo de giro azul, sea capaz de girar alrededor del cilindro que forma parte de la base del vehículo.

A la hora de estudiar este tipo de articulación cabe mencionar que cualquier posición o velocidad medida, siempre se referirá al frame conectado al nodo F con respecto del frame conectado al nodo B.

A continuación veremos algunas de las propiedades de mayor importancia de las que dispone este bloque:

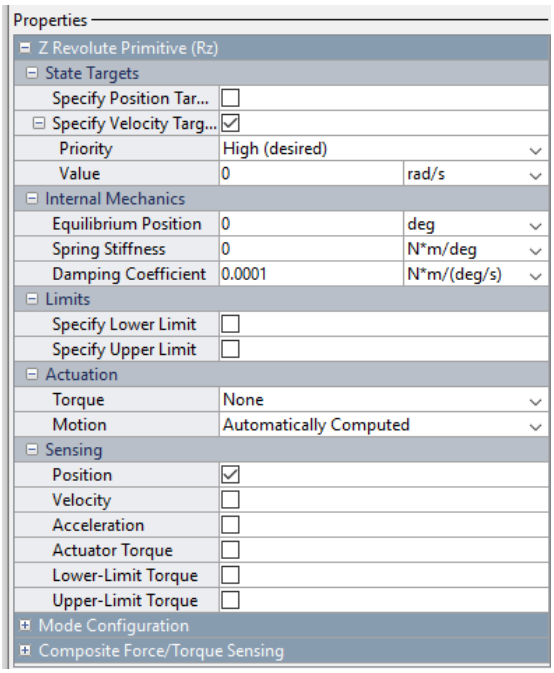


Ilustración 58: Propiedades junta de revolución

Las mas relevantes para este proyecto serán las de la categoría de 'Z Revolute Primitive', con lo que no entraremos en detalle sobre 'Mode configuration' y 'Composite Force/Torque Sensing'.

- State Targets: Esta opción nos permitirá definir estados iniciales de posición y/o velocidad al inicio de la simulación.
- Internal Mechanics: Define características mecánicas propias encontradas en articulación tales como el coeficiente de amortiguación existente entre ambas piezas.
- Limits: Permite limitar la velocidad y posición de un frame con respecto del otro de forma que se mantenga dentro de unos márgenes definidos
- Actuation: Permite actuar sobre la articulación, aplicando bien un par o indicando directamente el movimiento deseado sobre el mismo. Esto se aplicará al frame conectado al nodo F, de forma que si le damos una señal de 5 rad/s, el frame conectado al nodo F se moverá con una velocidad de 5 rad/s con respecto del frame conectado al nodo B.
- Sensing: Mediante esta opción podremos medir valores relativos al comportamiento de la articulación, tales como la velocidad del frame con respecto de la Base, o el par aplicado sobre la misma.

Spatial Contact Force

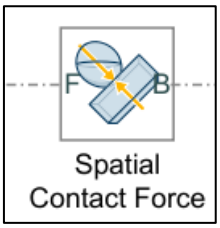


Ilustración 59: Bloque Spatial Contact Force

Este bloque es el que usaremos para poder modelizar el contacto de las ruedas del vehículo con el suelo sobre el que se desplaza. Para simular dicha fuerza de contacto, el bloque aplica 2 fuerzas entre los 2 objetos en contacto como podemos ver en la imagen:

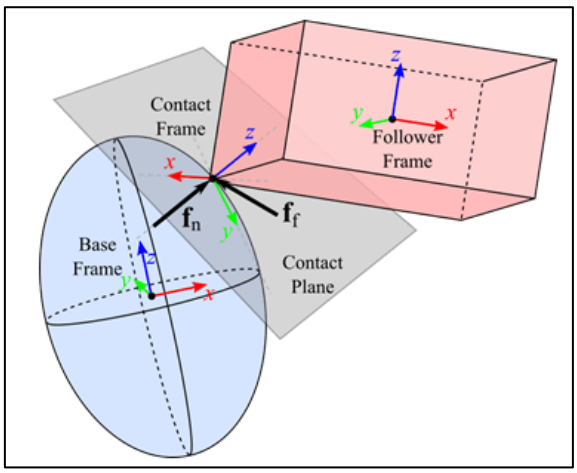
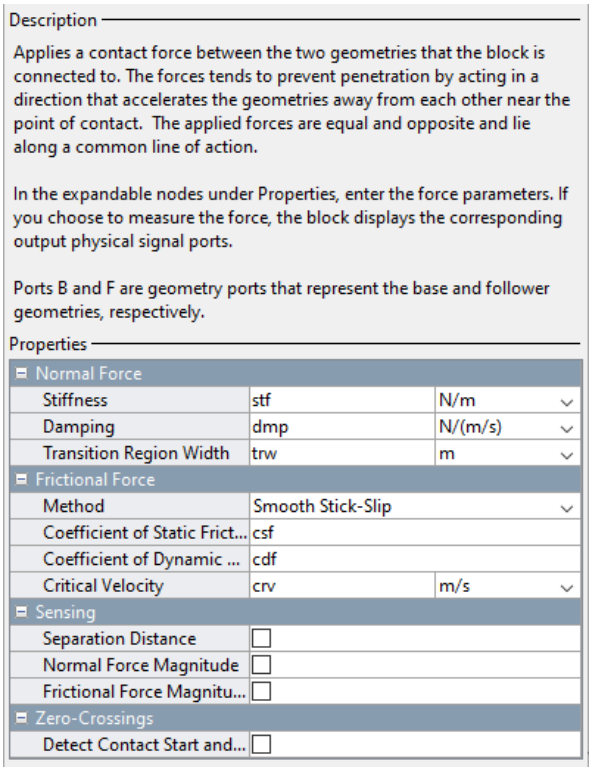


Ilustración 60: Fuerzas de contacto

Una fuerza f_n normal al plano de contacto de ambos cuerpos que se encarga de mantener separados ambos cuerpos y una fuerza f_f paralela al plano de contacto que representa la fuerza de fricción que se produce entre ambos modelos.

A continuación, veremos algunos de los parámetros que podremos modificar para definir adecuadamente dichas fuerzas descritas anteriormente.



Description

Applies a contact force between the two geometries that the block is connected to. The forces tends to prevent penetration by acting in a direction that accelerates the geometries away from each other near the point of contact. The applied forces are equal and opposite and lie along a common line of action.

In the expandable nodes under Properties, enter the force parameters. If you choose to measure the force, the block displays the corresponding output physical signal ports.

Ports B and F are geometry ports that represent the base and follower geometries, respectively.

Properties

Normal Force		
Stiffness	stf	N/m
Damping	dmp	N/(m/s)
Transition Region Width	trw	m
Frictional Force		
Method	Smooth Stick-Slip	
Coefficient of Static Frict...	csf	
Coefficient of Dynamic ...	cdf	
Critical Velocity	crv	m/s
Sensing		
Separation Distance	☐	
Normal Force Magnitude	☐	
Frictional Force Magnitu...	☐	
Zero-Crossings		
Detect Contact Start and...	☐	

Ilustración 61: Parámetros Spatial Contact Force

A la hora de definir el comportamiento de la fuerza normal entre ambos cuerpos Simulink considera a los dos cuerpos en contacto como un sistema masa-muelle-amortiguador, con lo que los parámetros a definir son los relativos a dicho sistema.

- Stiffness: Define la resistencia de contacto que ofrecen los cuerpos a introducirse el uno en el otro.
- Damping: Define el amortiguamiento que se produce entre las piezas al entrar en contacto.
- Transition Región Width: Región en el espacio en el que la fuerza producida por el sistema muelle-amortiguador alcanza su máximo valor.

Para simular la fricción que se produce entre ambos cuerpos, el bloque calcula dicha fuerza de fricción en función de la fuerza normal que existe entre ambos cuerpos y un coeficiente de fricción.

$$F_f = \mu \cdot N$$

Dicho coeficiente de fricción es el que definiremos en los parámetros del bloque:

- Method: Indica el método utilizado para obtener la fuerza de rozamiento entre ambos cuerpos.
- Coeficiente de Fricción Estática: Este parámetro indica el valor del coeficiente de rozamiento cuando la velocidad tangencial relativa entre ambos cuerpos se encuentra cercana a 0. Es decir, el coeficiente que se aplicaría cuando las ruedas se desplazan por el suelo sin deslizar.

- Coeficiente de fricción dinámica: Este coeficiente indica el coeficiente de rozamiento entre ambos cuerpos cuando la velocidad relativa entre ambos cuerpos supera el límite de velocidad crítica.
- Velocidad crítica: Velocidad que limita el punto a partir del cual empieza a hacer efecto el coeficiente de fricción dinámica.

El bloque permite realizar mediciones con respecto de los parámetros significantes que se han ido mencionando anteriormente tal y como la distancia entre ambos puntos de contacto, la fuerza normal entre ambos cuerpos y la fuerza de fricción producida en el punto de contacto.

3.2.2 Vehículo

A continuación, describiremos como se ha realizado el diseño del modelo de Simscape del vehículo. Se irán describiendo todos los componentes que forman el vehículo incluyendo los bloques que representan las uniones de los distintos componentes.

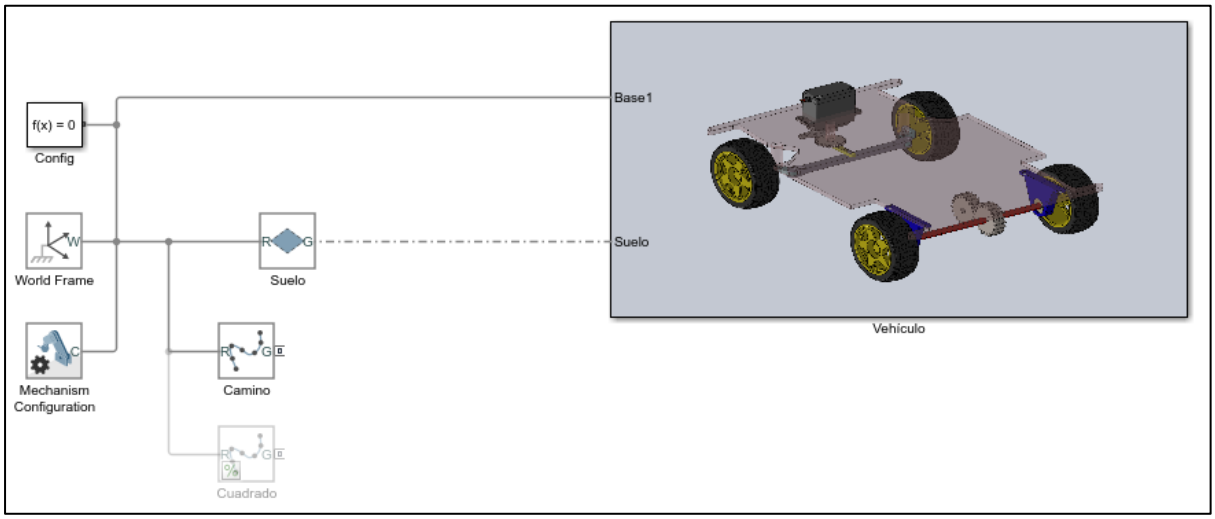


Ilustración 62: Modelo Simscape vehículo - Nivel superior

De la imagen del modelo completo de Simulink, puede apreciarse que en el nivel superior, el vehículo se encuentra conectado mediante una línea gris continua a los bloques generales del conjunto y mediante una línea discontinua al bloque que usaremos para modelizar el suelo sobre el que se apoyan las ruedas del vehículo.

Los 5 bloques del lado izquierdo del sistema tienen cada uno su propia función en el conjunto del vehículo:

- Un primer bloque '*Config*' que dictamina la configuración del Solver que utilizará MATLAB para realizar la simulación
- Un bloque '*World Frame*' que representa el origen del sistema mecánico que diseñaremos. Podría compararse con el concepto de conexión a tierra en una instalación eléctrica.
- Un bloque '*Mechanism Configuration*' que determina los parámetros mecánicos de simulación tal y como el valor de la aceleración de la gravedad

- El bloque denominado '*Camino*' cuya función será la de representar sobre el suelo la trayectoria que se pretende que siga el vehículo
- Un bloque '*Suelo*' que representa un plano infinito en el espacio sobre el cual el vehículo podrá apoyarse mediante fuerzas de contacto, representadas por las líneas discontinuas que se conectan al vehículo.

Estas fuerzas de contacto se modelizan mediante los bloques de 'Spatial Contact Force' que ya se describieron anteriormente.

Vehículo general

De la misma forma que en el apartado de diseño 3D, se han separado en subsistemas diferentes el sistema de dirección y el sistema de tracción. Esto se debe en gran parte al hecho de que ambos disponen de su propio controlador independiente. El sistema de tracción contará con un controlador PI que controle la velocidad lineal del vehículo mientras que el sistema de dirección tratará de dirigir el vehículo de forma que siga la trayectoria definida.

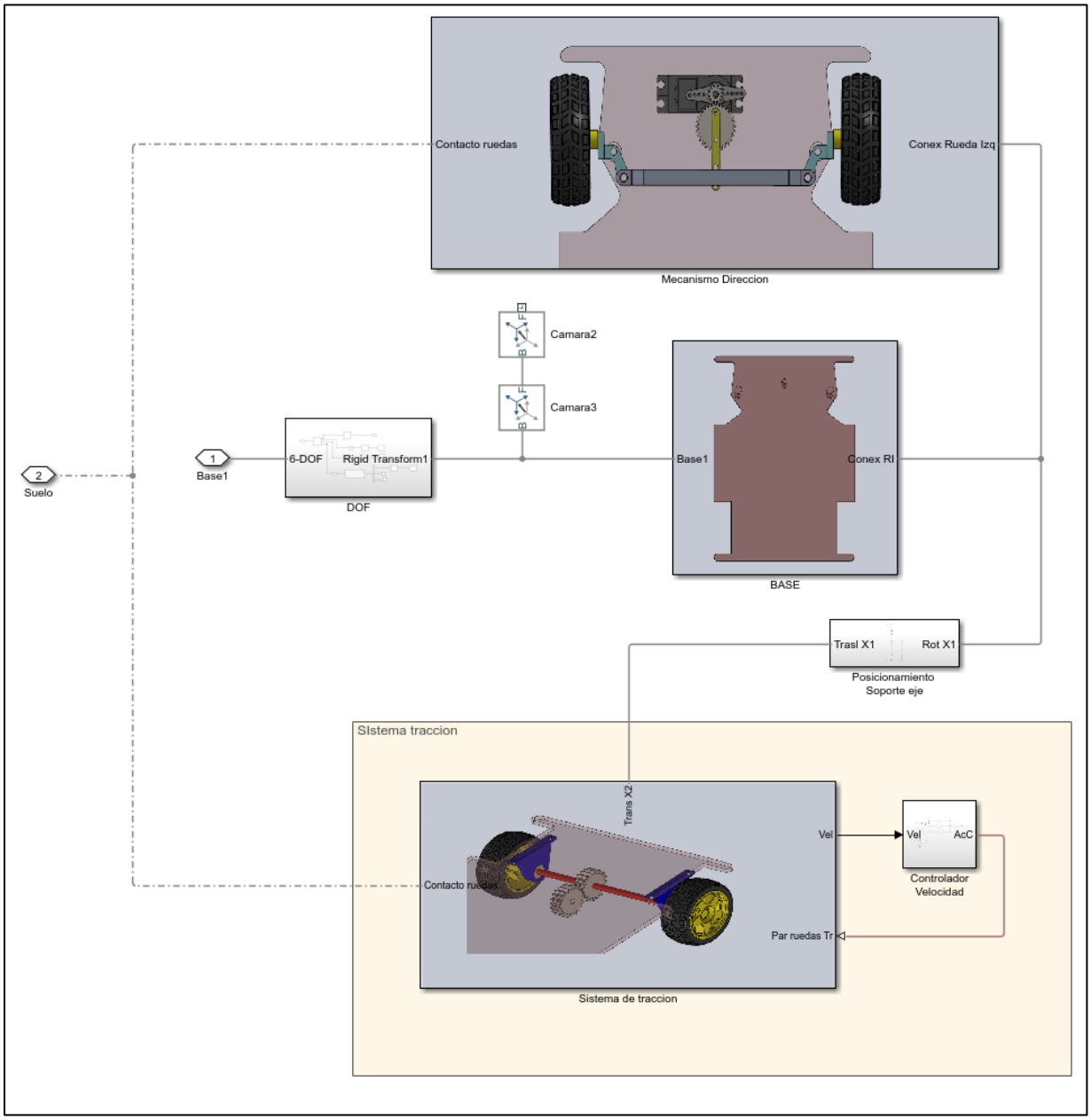


Ilustración 63: Modelo Simscape Vehículo

En este primer subnivel se encuentra la propia base del vehículo, así como algunos bloques necesarios para el correcto funcionamiento del modelo.

Base

En el centro del sistema encontramos la base del vehículo, la cual se introdujo brevemente a la hora de describir el Bloque 'Solid'.

Las cualidades más características de las que dispone dicho bloque son los frames que contiene, los cuales se pueden apreciar en la siguiente imagen:

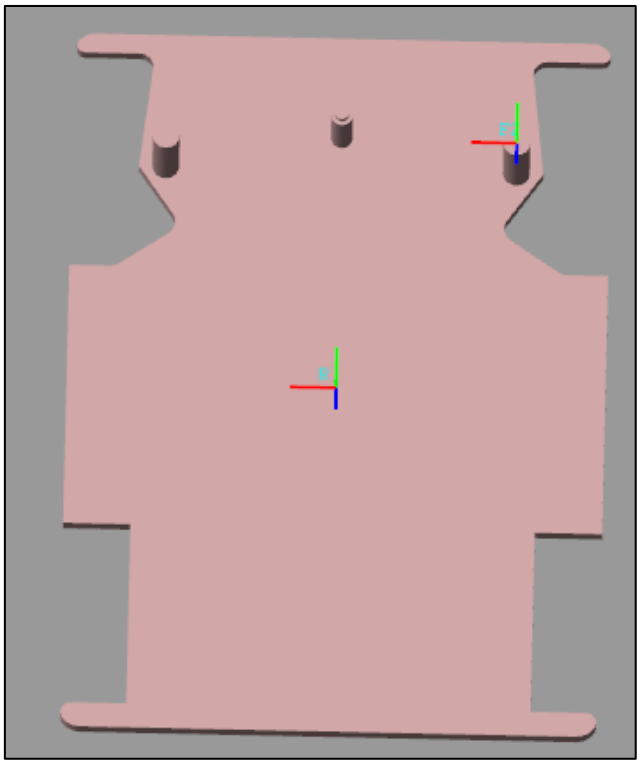


Ilustración 64: Frames Base

Vemos que dispone de dos frames:

- Un frame R, que por defecto se encuentra en el centro geométrico del solido importado
- Un frame F2 que se encuentra en el apoyo del lado derecho de la base (lado izquierdo si miramos a la base desde arriba) el cual usaremos para montar el sistema de dirección.

Conectado al frame R, tenemos el subconjunto DOF, que detallaremos a continuación, y 2 bloques de Rigid Transform los cuales utilizaremos más adelante para poder conectar una cámara durante la simulación.

DOF

Uno de los primeros Subconjuntos que describiremos es el subconjunto DOF. A continuación, veremos los distintos bloques que componen dicho subconjunto así como la función que desempeña dicho conjunto.

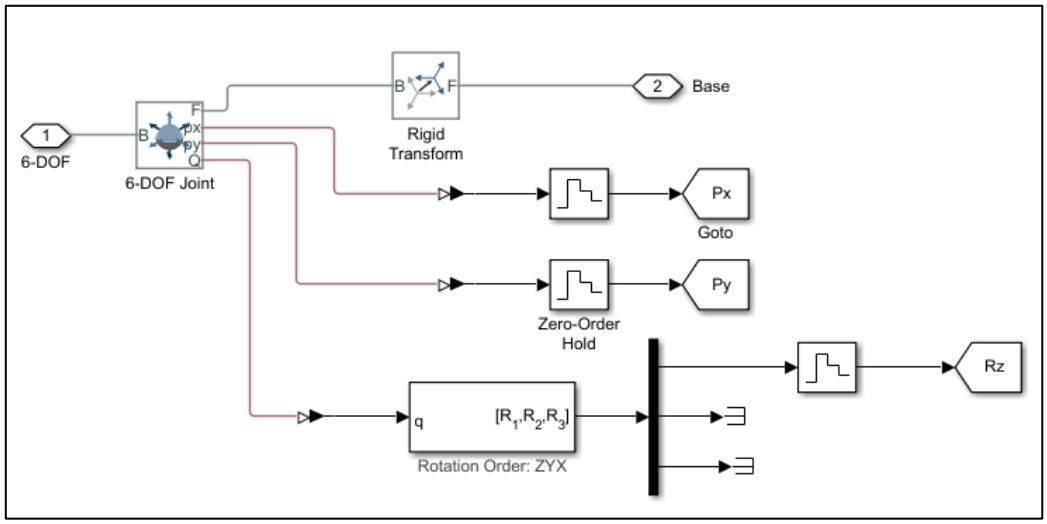


Ilustración 65: Subconjunto DOF

El bloque de mayor relevancia en este subconjunto es el bloque '6-DOF Joint' el cual representa una articulación de 6 grados de libertad. Esto quiere decir que el puerto que conectemos en el puerto F (Frame) no tendrá ningún tipo de restricción a la hora de desplazarse en el espacio. Esto permitirá que, al conectar el vehículo mediante cualquiera de sus puertos al puerto F de este bloque, el vehículo pueda moverse libremente por el plano.

El puerto 6-DOF de la parte izquierda del subsistema conecta directamente con el origen de coordenadas de la simulación, mientras que el puerto Base, conecta directamente con el sólido que representa la Base del vehículo.

Se conecta un bloque Rigid Transform entre el origen del sistema de simulación y el puerto del vehículo para orientar y posicionar adecuadamente el vehículo al inicio de la simulación. De otra forma podrían aparecer errores inesperados a la hora de realizar la simulación tal y como podemos ver en la siguiente imagen en el que las ruedas del vehículo atraviesan el suelo sobre el que deberían de apoyarse.

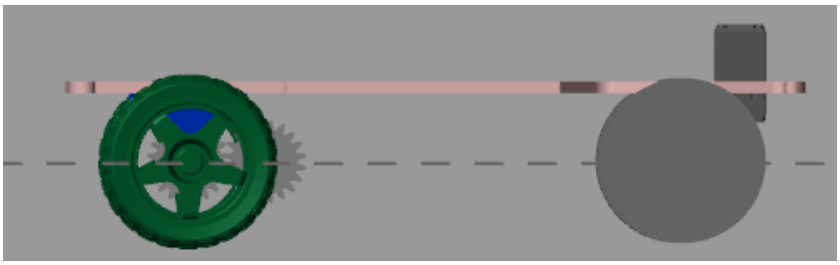


Ilustración 66: Problemática Simulación 1

Del bloque '6-DOF Joint' tomaremos las mediciones relativas a la posición del vehículo con respecto del centro de coordenadas. Tomaremos los datos de posición en el eje X/Y así como

el ángulo de inclinación del vehículo con respecto del eje X. Estas mediciones se observan en la siguiente imagen como Xo, Yo y φ:

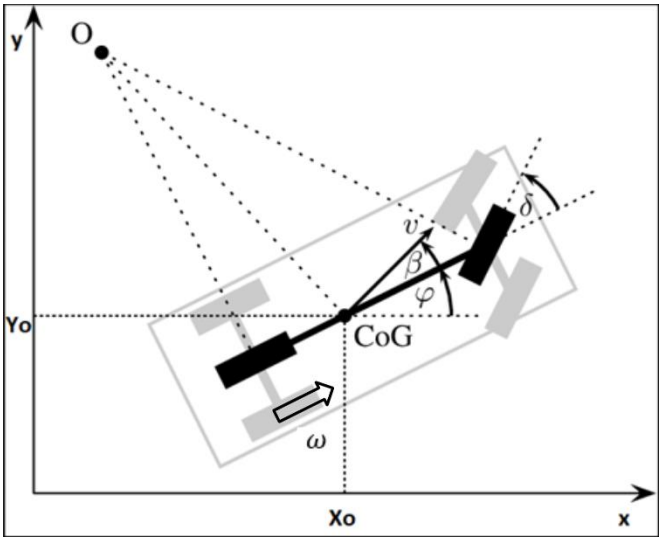


Ilustración 67: Modelo Vehículo

Debido a que las mediciones obtenidas del bloque '6-DOF Joint' resultan en señales físicas las cuales tienen asociadas unas unidades de medida tal y como rad/s o grados centígrados(º), es necesario realizar una conversión de estas unidades al sistema de Simulink. Esta conversión se realiza con el bloque PS-Simulink Converter:

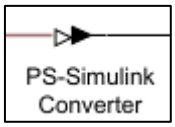


Ilustración 68: Bloque PS-Simulink

Además de dicha conversión, es necesario añadir un bloque 'Zero-Order hold' para asegurar el correcto funcionamiento de la medición en una simulación discreta y mantener un periodo fijo durante la simulación. Esto último es de gran utilidad ya que el microcontrolador que incorporemos al vehículo real dispondrá de un tiempo de computación mayor que el que podríamos utilizar en simulación.

Todas estas señales se envían mediante bloques 'Go To' Al subsistema de toma y muestra de datos y/o al Sistema de Control del vehículo.

Sistema de tracción

A continuación, describiremos como se ha diseñado el sistema de tracción en Simulink.

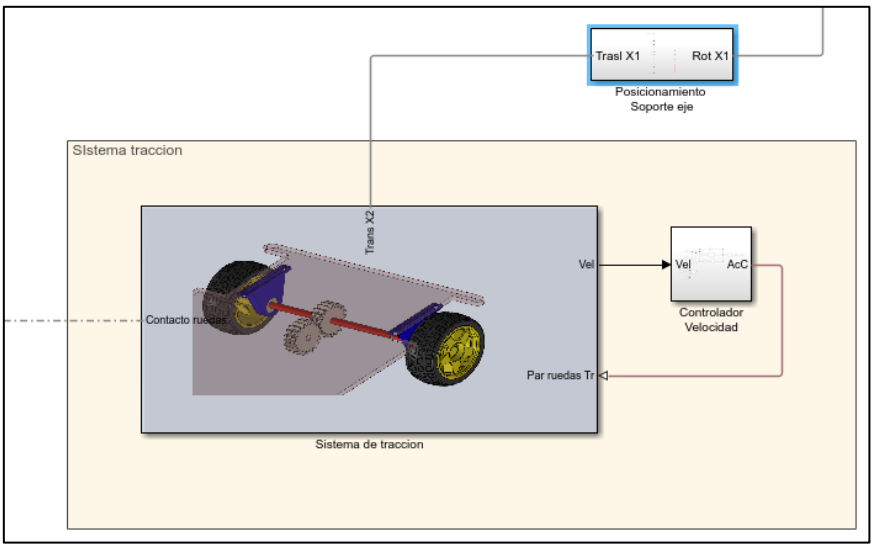


Ilustración 69: Sistema de tracción

Primeramente se observa un subconjunto denominado 'Posicionamiento Soporte eje'. Este subconjunto está formado por un conjunto de bloques Rigid Transform los cuales posicionan los soportes de los ejes traseros en función del apoyo del sistema de dirección que contiene la Base del vehículo.

El propio sistema de tracción contiene a su vez un controlador PI de velocidad que usaremos para controlar la velocidad del vehículo.

Este controlador está compuesto por un bucle de control cerrado el cual mide la velocidad de las ruedas del vehículo y en función del valor deseado, envía una señal del par a aplicar a las ruedas para poder alcanzar dicha velocidad deseada. En la siguiente imagen se puede observar el controlador diseñado:

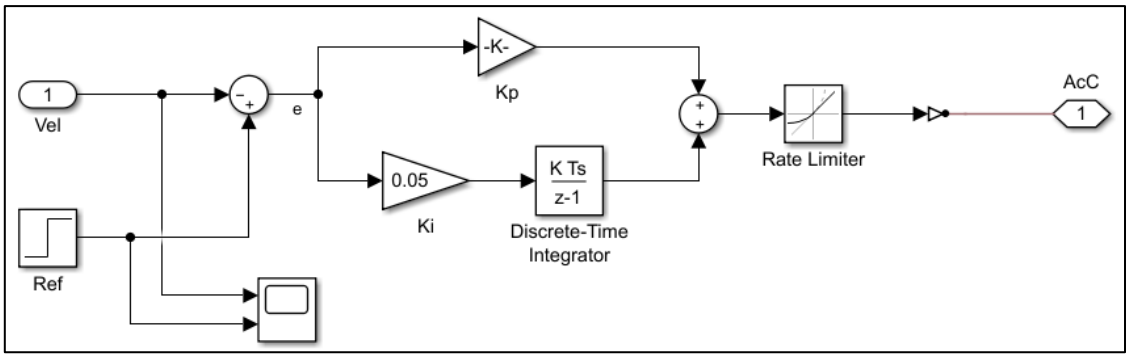


Ilustración 70: Controlador PI velocidad

Este controlador toma el valor de velocidad leído por el sensor y le resta a una referencia dicho valor, obteniendo con ello el valor del error entre la velocidad real de las ruedas del vehículo y el valor de velocidad deseado. A dicho error le aplicamos por un lado una constante proporcional Kp para obtener la acción de control proporcional y por otro lado le aplicamos un

integrador para obtener el valor de la acción integral que le aplicaremos al sistema de tracción. Sumando ambas componentes obtenemos el valor de la acción de control que requiere nuestro vehículo.

Para evitar cambios bruscos y problemas de simulación, se ha añadido un bloque 'Rate Limiter' que limita la pendiente máxima que puede tener el par aplicado a las ruedas.

En el prototipo real que se pretende diseñar, la acción de control calculada, correspondería al voltaje que deberíamos aplicarle al motor eléctrico para girar el eje de las ruedas. En esta simulación en cambio, dicha acción de control representa el par que aplicaremos directamente sobre el engrane que conectaría con el motor del vehículo. Esto lo veremos con más claridad al detallar la modelización del propio sistema de dirección en Simulink.

Sistema de tracción Simulink

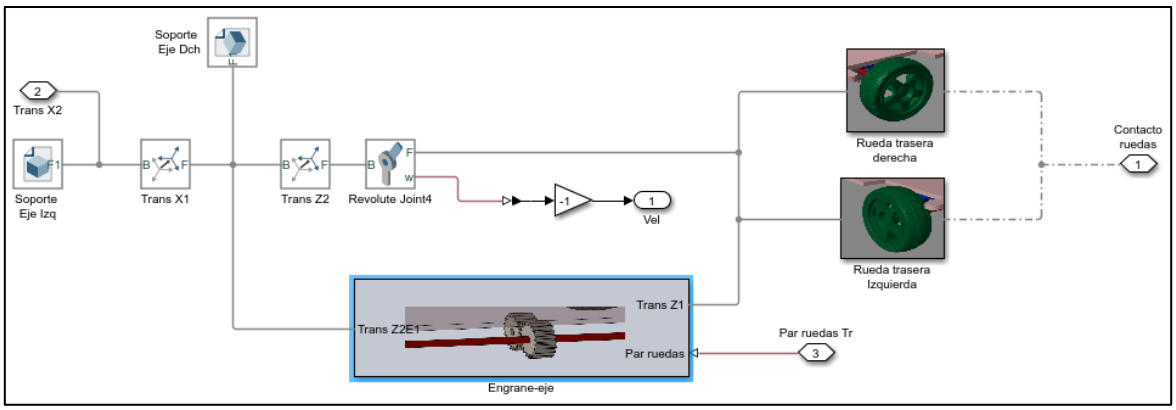


Ilustración 71: Sistema de tracción Simulink

En la imagen anterior se puede apreciar el modelo realizado en Simulink para poder simular adecuadamente el funcionamiento del sistema de tracción del vehículo. En la siguiente imagen veremos a modo de recordatorio, el sistema mecánico que se pretende diseñar:

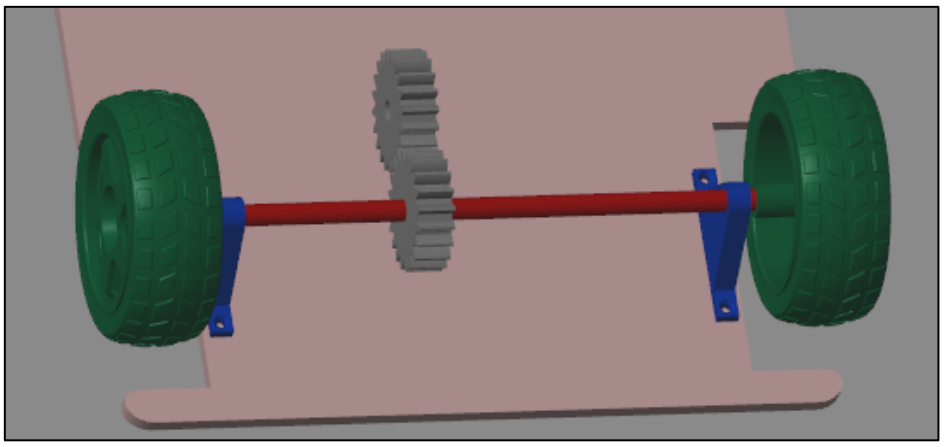


Ilustración 72: Sistema de tracción 3D

Dicho subsistema se compone de 2 ruedas motrices montadas sobre un eje giratorio el cual está posicionado sobre unos soportes como se ve en la imagen. Dicho eje estará impulsado por el par de engranes de la imagen.

Para representar dicho conjunto mecánico se ha decidido partir desde los soportes del motor, como podemos ver en el diagrama de Simulink superior. Tras posicionar el soporte izquierdo podremos posicionar el resto de componentes del sistema de tracción, posicionando primero el soporte derecho mediante un bloque Rigid Transform.

En la siguiente imagen podemos ver el subsistema Engranaje-eje, que se encarga de otorgarle el giro necesario a las ruedas para que el vehículo pueda desplazarse.

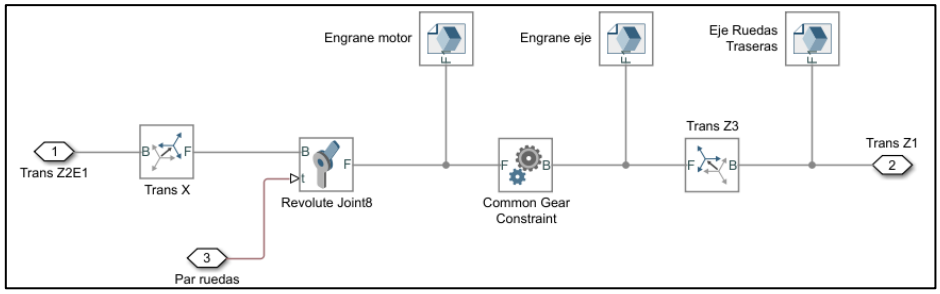


Ilustración 73: Subsistema Engranaje-eje

Dicho subsistema está montado de forma que el engrane del motor se encuentre desplazado una cierta distancia del soporte que permita que engrane con el engranaje montado sobre el eje.

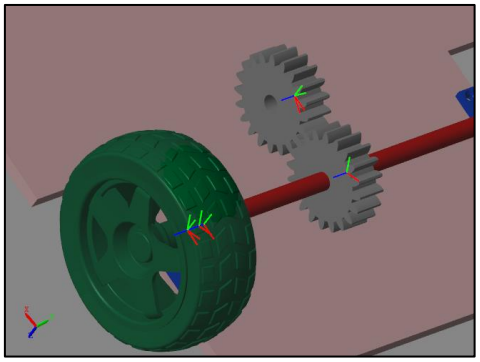


Ilustración 74: Montaje engranaje

Sobre el desplazamiento realizado añadimos un bloque de Revolute Joint que permite que el engranaje sea capaz de girar sobre su eje. Es en esta articulación en la que le aplicamos la fuerza motriz que moverá nuestro vehículo. Para ello le aplicamos el valor de par obtenido en el sistema de control al bloque de Revolute Joint que conecta al Engrane Motor.

Este movimiento se transmite al engranaje montado sobre el eje y al propio eje mediante el bloque de 'Common Gear Constraint' que simula la interconexión entre 2 engranajes, definiendo la distancia entre centros de engrane y la relación de transmisión entre los engranes simulados.

Las ruedas del vehículo y el eje las conectaremos de forma que giren alrededor del agujero que une ambos soportes. Para ello se conectan tanto las ruedas como el eje del vehículo a una articulación de revolución, permitiendo con ello que giren conjuntamente alrededor del soporte del vehículo.

En la siguiente imagen puede apreciarse el subconjunto que modeliza las ruedas traseras del vehículo:

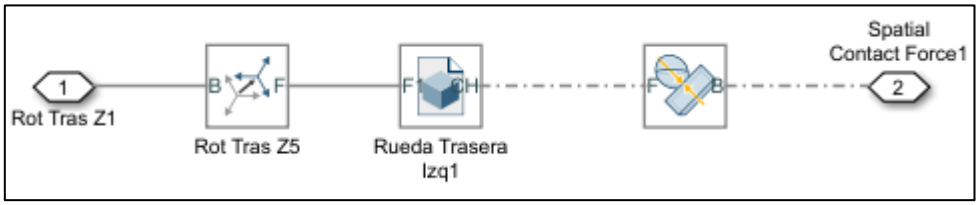


Ilustración 75: Modelo rueda trasera

El subsistema es el mismo para ambas ruedas traseras. Dicho subsistema se encuentra conectado de forma rígida al eje trasero y mediante una articulación de revolución al soporte del eje, permitiendo con ello el giro simultaneo de las piezas correspondientes.

Para modelizar cada rueda se ha incluido primeramente un bloque de Rigid Transform, que posiciona a la rueda en su lugar adecuado en el eje. Seguidamente se conecta un bloque Solid, el cual contendrá la geometría 3D de la propia rueda que pretendemos simular, junto a un bloque Spatial Contact Force, el cual nos permitirá que las ruedas puedan apoyarse sobre el suelo en la simulación. El bloque Spatial Contact Force conecta directamente con el bloque Suelo que se describió en el apartado de Vehículo General.

Sistema de dirección

En el siguiente apartado describiremos como se ha modelizado el sistema de dirección que nos permite dirigir el vehículo durante la simulación. En la siguiente imagen puede apreciarse el subsistema de dirección en el nivel superior del vehículo. Esta imagen nos sirve además a modo de recordatorio de los componentes que componen el sistema de dirección.

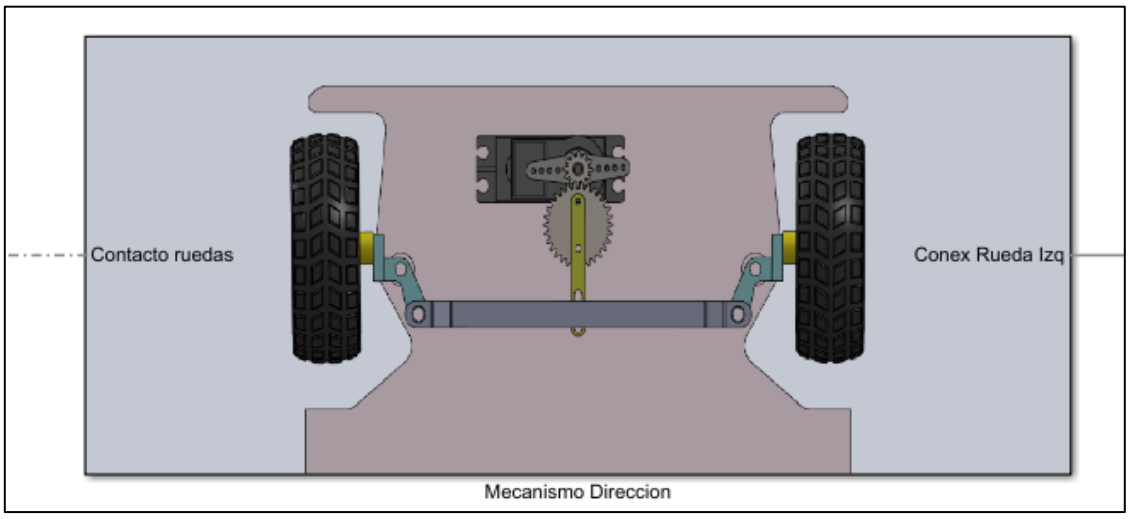


Ilustración 76: Mecanismo dirección

El subsistema dispone de 2 nodos que conectan con el subsistema general: El contacto de las ruedas con el suelo y el posicionamiento del mecanismo de dirección sobre la base del vehículo.

En la imagen inferior se aprecian los distintos subensamblajes que componen el sistema de dirección del vehículo:

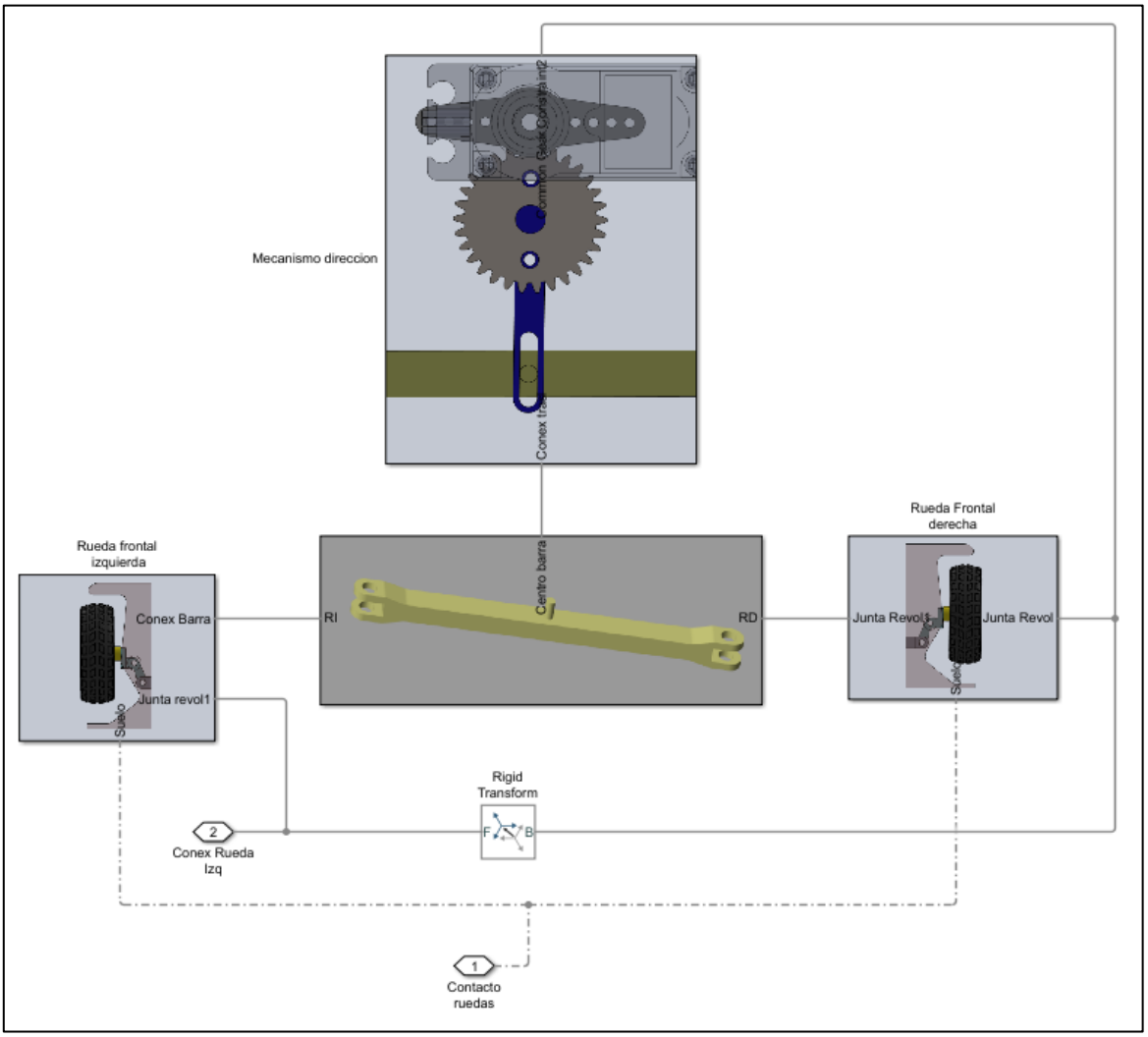


Ilustración 77: Subsistema dirección

Se pueden apreciar 4 subensamblajes claramente diferenciados:

- Conjunto rueda izquierda/derecha
- Barra central que transmite el movimiento del Servomotor a las ruedas directrices
- Mecanismo de dirección el cual produce y transmite el giro del servo a la barra que une las ruedas del vehículo.

El conjunto rueda modeliza la propia rueda directriz junto al componente encargado de transmitir el movimiento de la barra a las ruedas(bieleta de dirección). En la siguiente imagen pueden verse los componentes exactos que se modelizan:

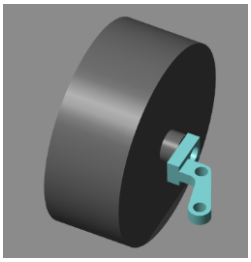


Ilustración 78: Rueda izquierda

Se trata por un lado del conector rueda-barra mencionado anteriormente, así como la propia rueda, la cual se ha decidido modelizar mediante un cilindro con las mismas dimensiones que la rueda, con el fin de reducir el tiempo de simulación.

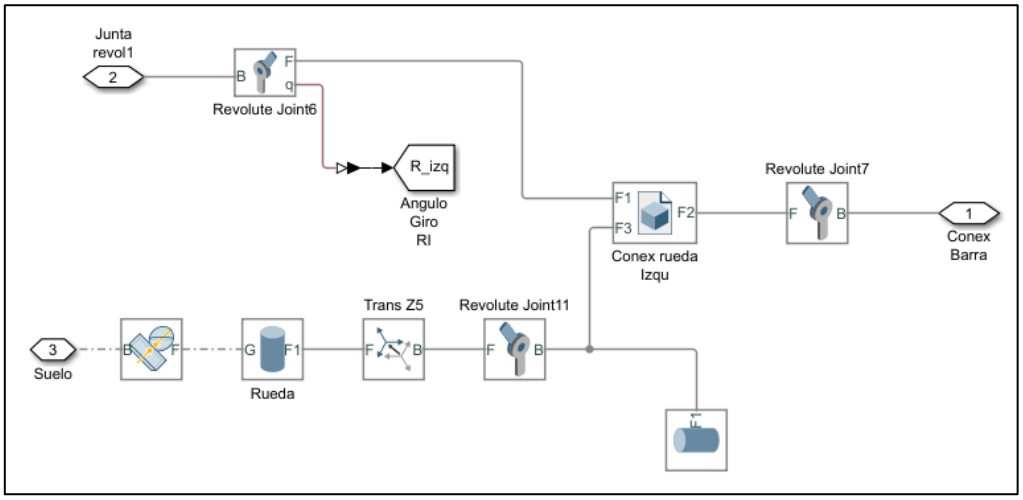


Ilustración 79: Diagrama rueda directriz

El diagrama contiene un total de 3 conexiones con el nivel de subensamblaje superior siendo estas:

- El contacto de las ruedas con el suelo, modelizado mediante el bloque de Spatial Contact Force
- La conexión del Conjunto a la base del vehículo, modelizada mediante la articulación de revolución 'Revolute Joint6'
- La unión de este sistema con la barra de control modelizada mediante la articulación de revolución 'Revolute Joint7'

Como puede verse en el diagrama, el componente principal de este subconjunto podría considerarse la bieleta de dirección que sirve de unión entre base, rueda y barra de dirección.

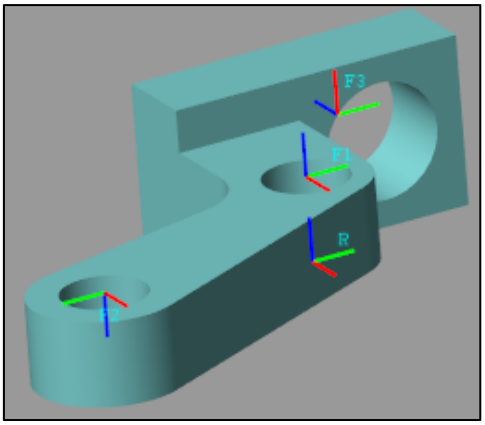


Ilustración 80: Bieleta de dirección

Esta está compuesta por un total de 4 frames, siendo uno el Frame R incluido por defecto. Cada frame está colocado de forma que permita unirse correctamente a los elementos que componen el sistema de dirección tal y como el centro del apoyo de la base o el extremo de la barra de dirección.

El frame F1, es el que como podemos ver en el diagrama, conecta mediante una articulación de revolución con la base del vehículo. Mediante esta articulación podremos medir el ángulo de giro de la rueda directriz a la que se encuentre conectado la bieleta.

El frame F3 se conecta a una articulación de revolución que conecta con la rueda. Esto permite que la rueda pueda girar alrededor del eje Z de dicho frame.

La barra de dirección del subsistema de dirección solo incluye un bloque Solid con la propia barra de dirección. En la imagen inferior pueden verse los frames que contiene dicho sólido.

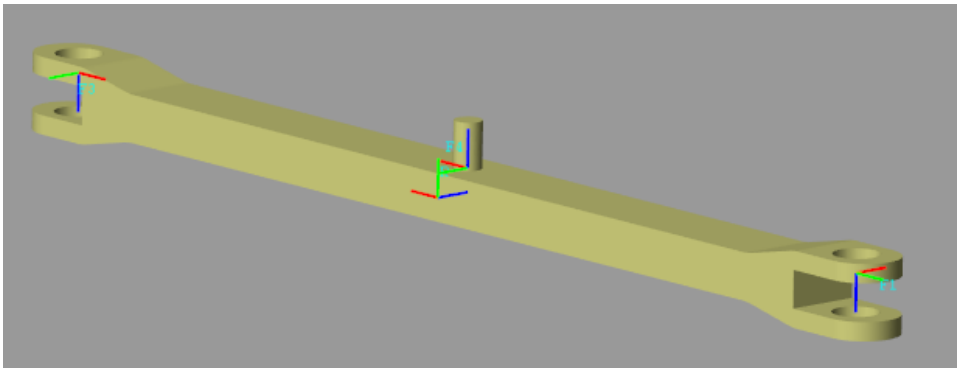


Ilustración 81: Barra dirección

La barra dispone de un frame en cada punta que permite conectarse con las bieletas de dirección además de un frame en el cilindro central que permite que se conecte con el brazo del mecanismo de dirección que veremos a continuación.

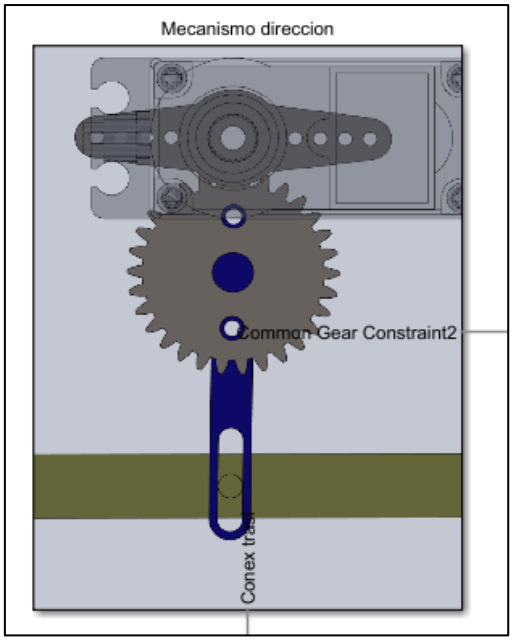


Ilustración 82: Mecanismo de dirección

Este mecanismo se compone principalmente del Servomotor que proporcionará el movimiento de giro de las ruedas, así como el brazo de dirección (pieza de color azul) que transmite el giro del servo a la barra de dirección. En la siguiente imagen puede verse el diagrama de Simulink diseñado para simular este mecanismo:

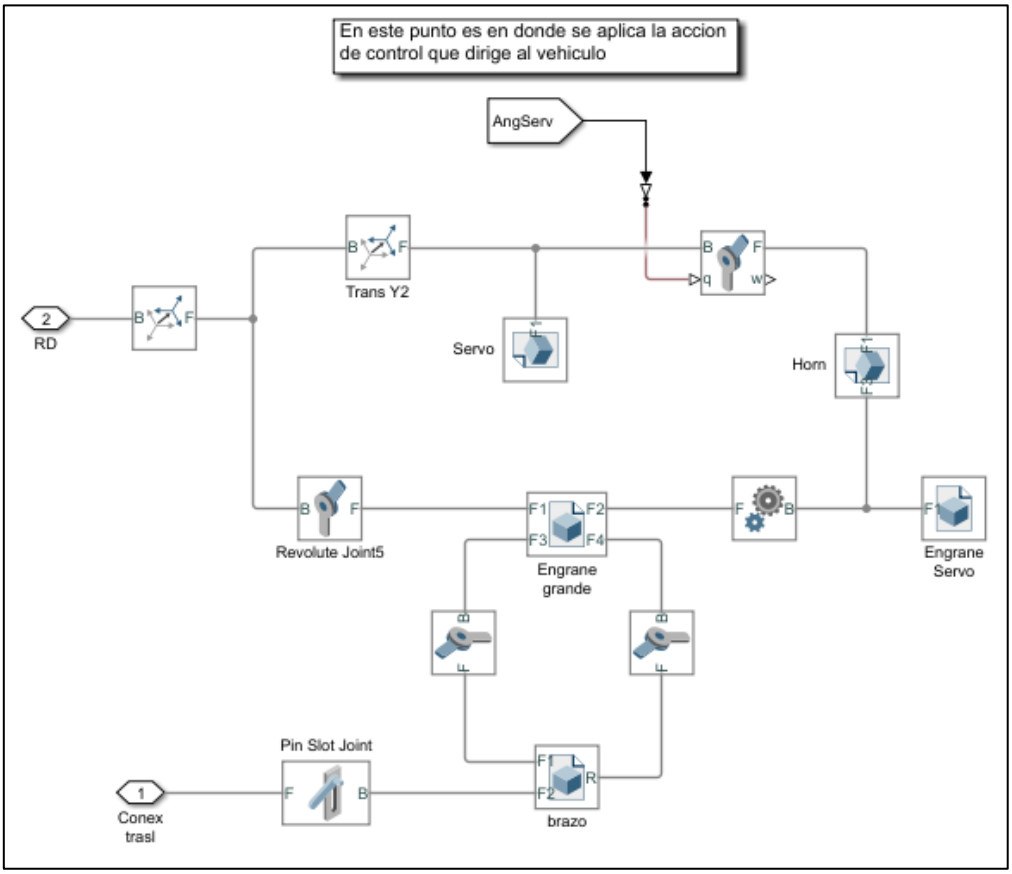


Ilustración 83: Diagrama mecanismo de dirección

Este subsistema está compuesto por varios componentes y podría considerarse el subsistema más complejo del vehículo. Desde el nodo RD posicionamos correctamente todas las piezas que componen el ensamblaje mediante bloques de Rigid Transform.

Desde la parte superior del diagrama al inferior nos encontramos primeramente con la articulación de revolución que une el servomotor con el brazo del mismo. Es en esta articulación en donde se aplica la señal de giro obtenida en el sistema de Control de dirección que describiremos más adelante.

El giro de esta articulación se transmite al engrane grande mediante el bloque de Common Gear Constraint. A modo aclaratorio se menciona que el bloque de Engrane servo no es el bloque que transmite realmente el movimiento del servo al bloque Engrane Grande, sino que su función podría considerarse meramente estética.

La conexión del Engrane Grande con el brazo del mecanismo de dirección se realiza a través de 2 articulaciones de revolución. El objetivo es que dichas articulaciones simulen de forma aproximada el montaje real en el que como vemos en la imagen inferior:

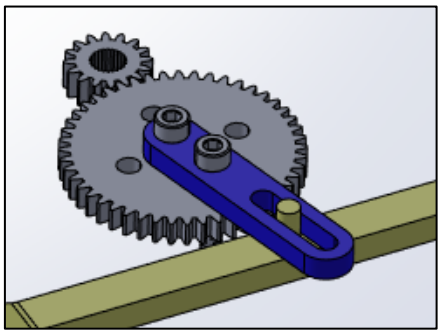


Ilustración 84: Montaje engrane-brazo

En este montaje el brazo de dirección se mueve conjuntamente con el Engranaje.

Finalmente para modelizar la unión del brazo de dirección con la barra principal de dirección, se ha utilizado un bloque 'Pin Slot Joint' el cual representa una articulación de tipo rotación-traslación, permitiendo que el saliente de la barra de dirección se desplace a lo largo del hueco que dispone el brazo sin la necesidad de que los componentes mantengan el mismo ángulo de movimiento.

3.2.3 Toma y muestra de datos

A continuación veremos como se ha realizado la toma de datos a la hora de simular el vehículo y las variables que se han decidido medir. Conviene mencionar que durante la simulación podremos medir casi cualquier variable involucrada en el movimiento del vehículo, algo que en el montaje real se verá enormemente reducido debido a la multitud de sensores que se requeriría:

Los datos que se han decidido medir son los siguientes:

- Posición (x, y) del vehículo en el plano
- Velocidad de giro del eje trasero
- Ángulo de giro de ambas ruedas
- Par aplicado al motor trasero
- Ángulo de giro del vehículo con respecto del eje X
- Velocidad real del vehículo
- Velocidad de giro del vehículo completo

En base a dichas variables podremos calcular otras variables tal y como el radio de giro del vehículo.

En la siguiente imagen podemos ver el conjunto de datos que se toman durante la simulación:

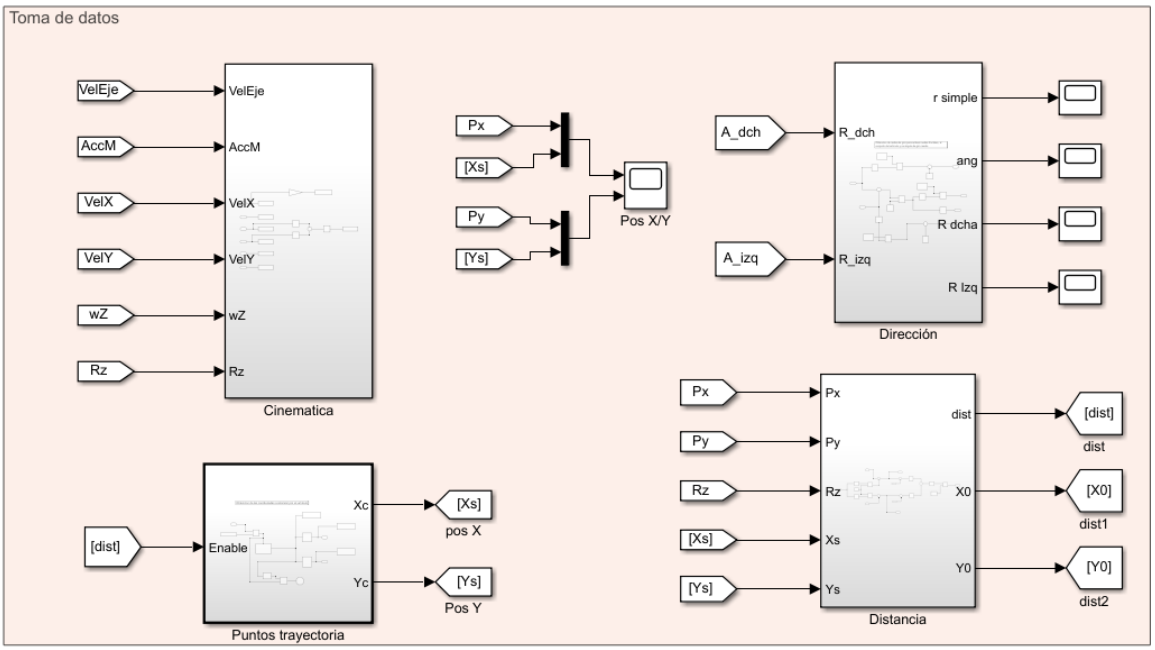


Ilustración 85: Toma de datos Simulink

Esta imagen denota los 5 grupos de medida que se han realizado para la toma de datos durante la simulación. Los 3 grupos superiores analizan el funcionamiento del vehículo durante la simulación, mientras que los 2 inferiores se encargan del adecuado funcionamiento del sistema de dirección autónoma.

Los grupos se distinguen en:

- Análisis cinemático del vehículo
- Posición del vehículo durante la simulación
- Variables relacionadas con la dirección del vehículo.
- Obtención de la trayectoria a seguir por el vehículo
- Distancia del vehículo al siguiente punto de control

Cinemática

De este conjunto se obtienen todas las variables relacionadas con la cinemática del vehículo tal y como la velocidad de desplazamiento del vehículo o la velocidad de giro del vehículo.

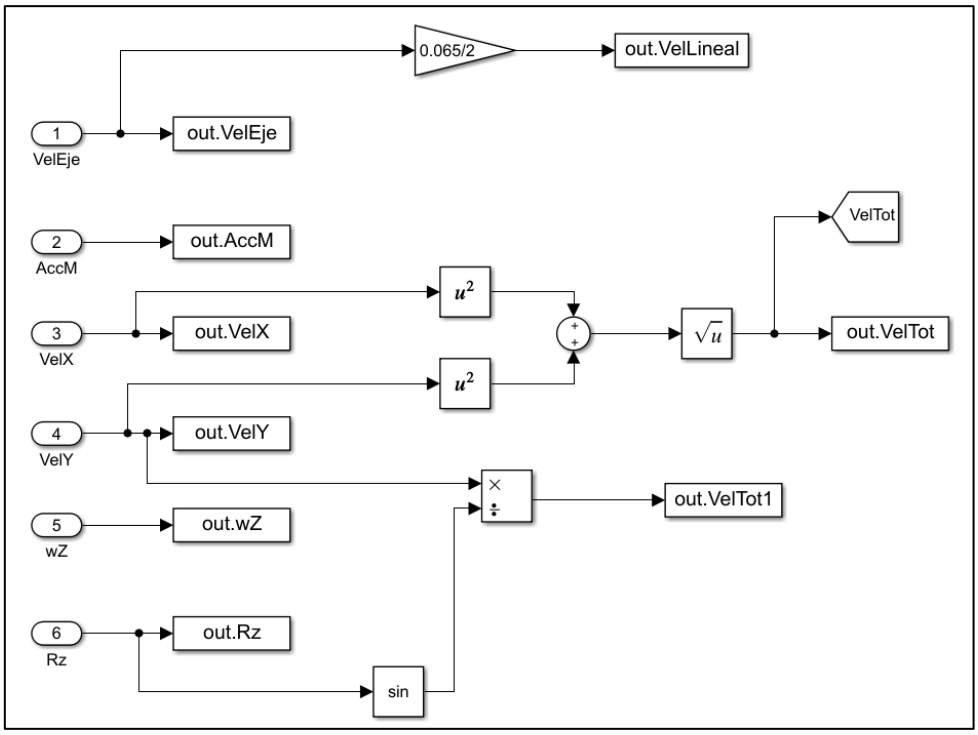


Ilustración 86: Datos cinematica

Posición X-Y

Este conjunto representa la trayectoria de puntos que debe seguir el vehículo (Xs,Ys), con respecto a la trayectoria que realmente sigue (Px,Py).

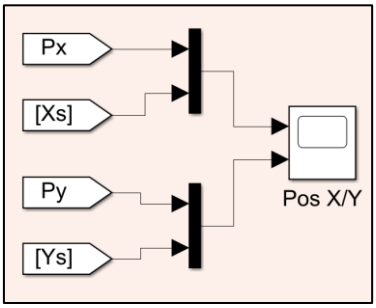


Ilustración 87: Posición X-Y vehículo

Dirección

Este subconjunto procesa todos los datos relativos a la dirección del vehículo. La función de este subconjunto es la de mostrar los radios de giro del vehículo, así como los ángulos de giro de las ruedas al girar el vehículo.

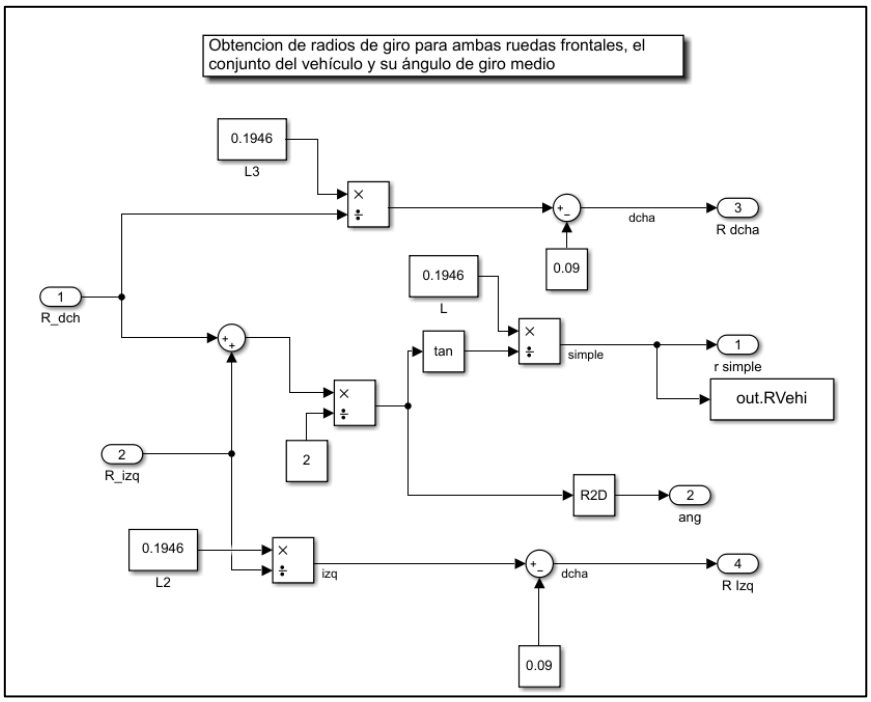


Ilustración 88: Toma datos dirección

Se obtienen los radios de giro del vehículo en base a la geometría constructiva del vehículo.

Puntos trayectoria

La función de este grupo es la de indicarle al resto del sistema de control cual es el punto en el espacio que debe alcanzar el vehículo.

Esto se ha logrado definiendo un sistema de control basado en un vector de coordenadas que debe seguir el vehículo. El vehículo tratará de alcanzar un cierto punto de dicho vector para poder pasar al siguiente punto que compone la trayectoria a seguir. Como criterio que define que el coche se encuentre lo suficientemente cercano al punto deseado, se ha calculado la distancia del centro del eje trasero del vehículo al siguiente punto que debe alcanzar. Si dicha distancia es menor al diámetro de la rueda del vehículo, pasamos al siguiente punto. Este concepto se puede apreciar en las siguientes imágenes:

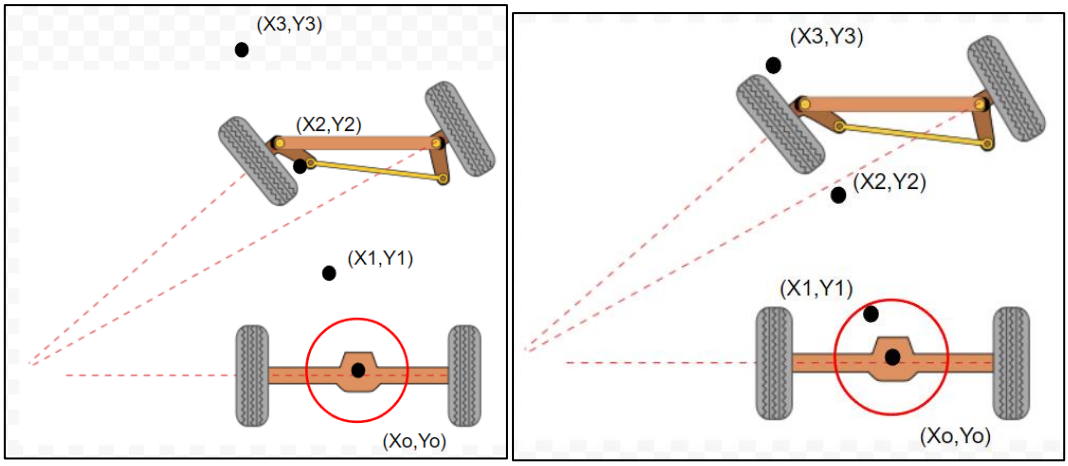


Ilustración 89: Puntos trayectoria

Una vez la coordenada (X1,Y1) se encuentre lo suficientemente cercano a (Xo, Yo) el vehículo pasará a tratar de alcanzar el siguiente punto (X2,Y2).

Para lograr esto se ha definido el siguiente sistema de control en Simulink:

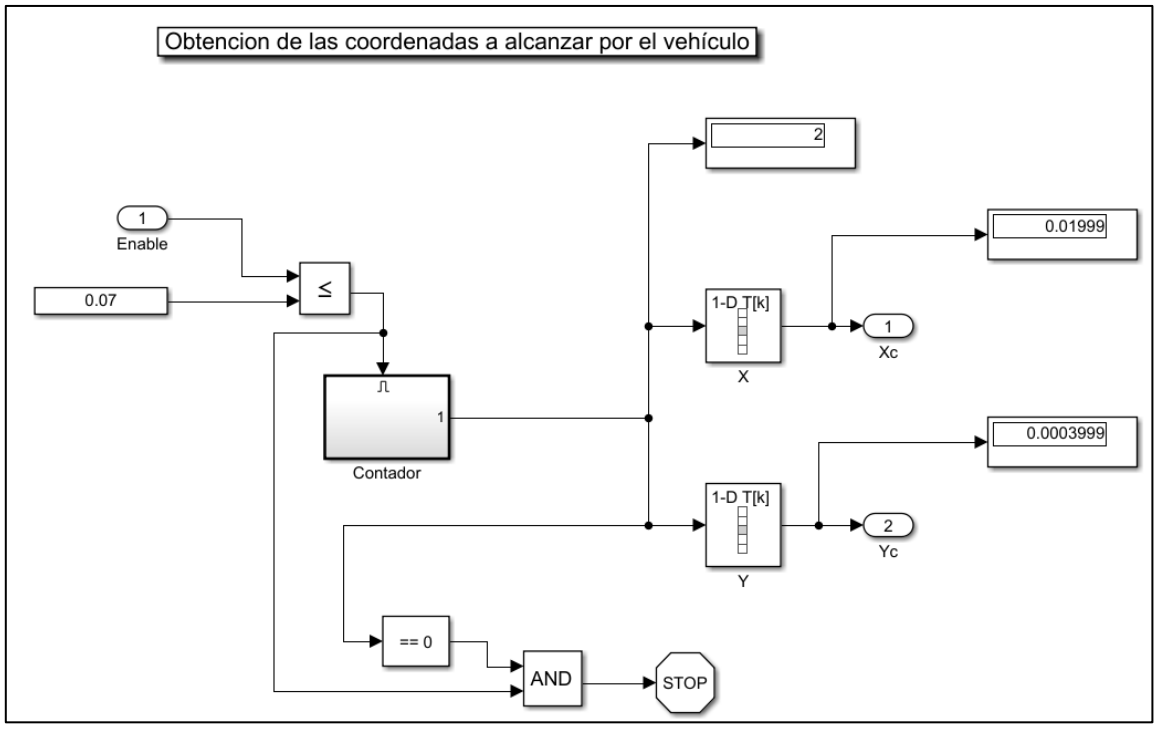


Ilustración 90: Obtención puntos trayectoria

Si recordamos del conjunto superior, recordaremos que el subsistema dispone de una sola entrada y 2 salidas:

- Distancia de las coordenadas del vehículo al siguiente punto que debe alcanzar (Distancia de control)
- Las coordenadas Xc-Yc que definen el punto al que debe dirigirse el vehículo.

El subsistema dispone de un Contador, el cual aumentará la cuenta cada vez que la distancia de control se encuentre por debajo de cierto valor.

Se incluye en la parte inferior un conjunto de bloques que terminará la simulación una vez se haya logrado pasar por todos los puntos incluidos en la trayectoria.

La parte derecha del sistema incluye los vectores de coordenadas X-Y que le indican al sistema de control el punto que debe alcanzar el vehículo

Cálculo distancia

En este grupo se trata de calcular la distancia existente entre las coordenadas del vehículo y las coordenadas del punto que debe alcanzar el vehículo.

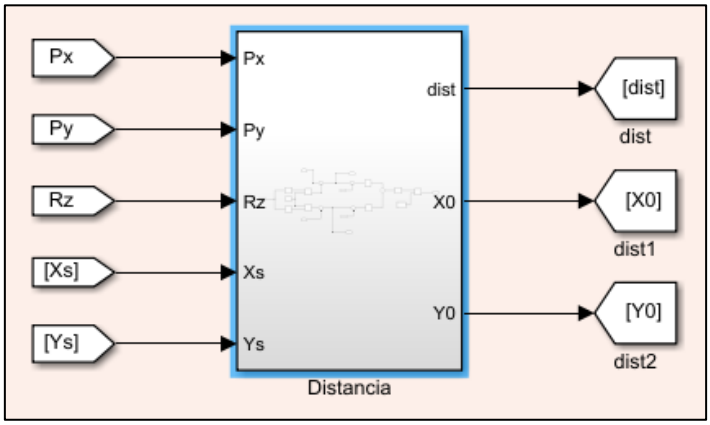


Ilustración 91: Cálculo distancia

El subconjunto dispone de un total de 5 entradas:

- Coordenada en el eje X del centro del vehículo, Px
- Coordenada en el eje Y del centro del vehículo, Py
- Orientación del vehículo con respecto al eje X, Rz
- Posición en el eje X del punto que debe alcanzar el vehículo Xs
- Posición en el eje Y del punto que debe alcanzar el vehículo Ys

Mediante las cuales obtiene 3 salidas:

- Distancia de control
- Coordenadas X-Y del eje trasero del vehículo

A modo aclaratorio se describirá primeramente el método utilizado para obtener la Distancia de control así como las coordenadas X-Y del eje trasero del vehículo.

Comenzaremos obteniendo las coordenadas del eje trasero del vehículo, basándonos en la siguiente imagen:

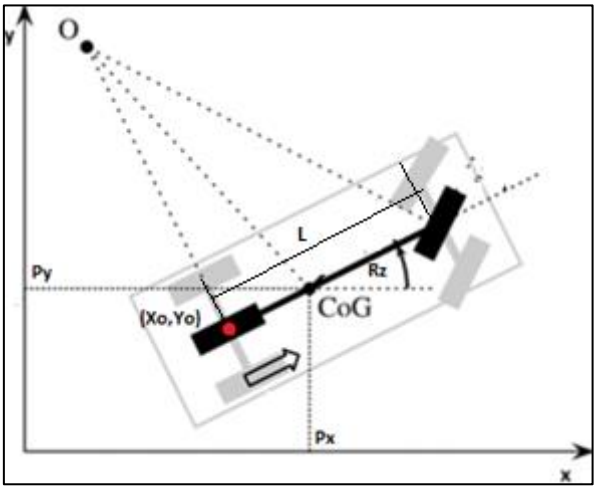


Ilustración 92: Coordenadas vehículo de Ackermann

En el cuál la variable L indica la distancia entre ejes del vehículo. Este dibujo lo podremos simplificar de la siguiente forma:

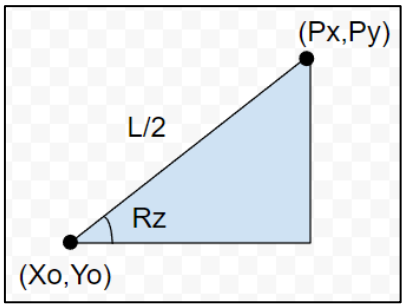


Ilustración 93: Cálculo coordenadas vehículo

En esta imagen se observa que, para obtener las coordenadas del eje trasero del vehículo, es suficiente con realizar un sencillo cálculo trigonométrico:

$$\cos(Rz) = \frac{Px - Xo}{\frac{L}{2}} \rightarrow Xo = Px - \frac{L}{2} \cdot \cos(Rz)$$

$$\sin(Rz) = \frac{Py - Yo}{\frac{L}{2}} \rightarrow Yo = Py - \frac{L}{2} \cdot \sin(Rz)$$

Una vez hemos obtenido las coordenadas del eje trasero podremos calcular la distancia de Control como el módulo del vector formado por el punto objetivo y el centro del eje trasero:

$$\sqrt{(Xo - Xs)^2 + (Yo - Ys)} = dist$$

En el siguiente diagrama de Simulink se puede ver el montaje realizado para obtener dichos valores:

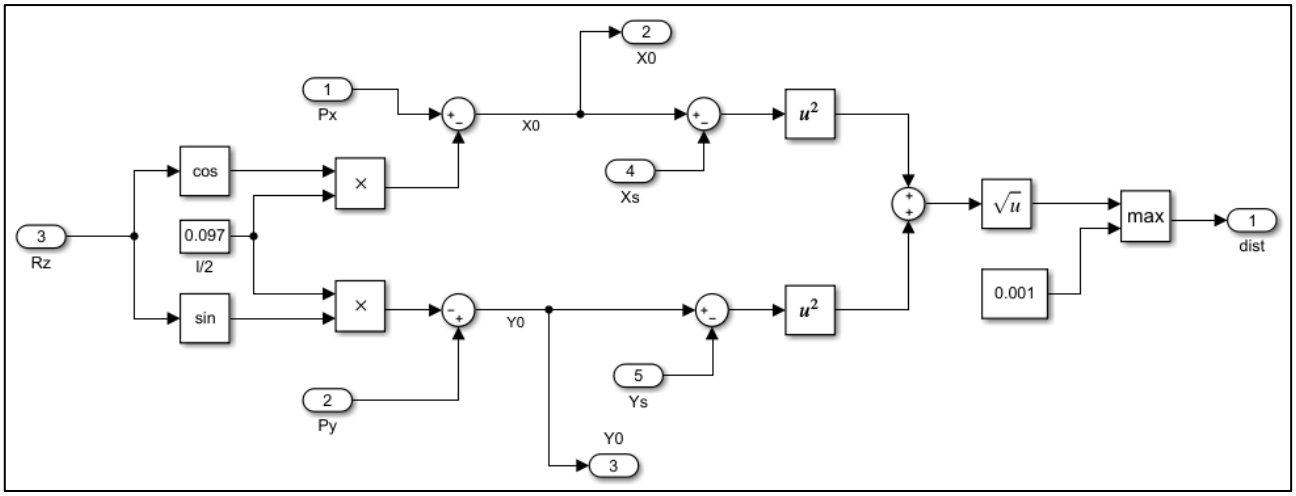


Ilustración 94: Diagrama cálculo distancia

En este diagrama se puede ver como al principio del mismo se utiliza el mismo para obtener las coordenadas Xo e Yo, tomando en consideración que la mitad del largo del vehículo se encuentra en 97 mm. Tras obtener dichas coordenadas se procede a obtener finalmente la distancia del punto objetivo con el centro trasero del vehículo.

3.2.4 Trayectorias

A la hora de diseñar el sistema de control para un vehículo de Ackermann conviene tomar en cuenta las limitaciones a la hora de girar que presenta el vehículo. Por ello se plantean 3 tipos distintos de trayectoria que se pretende que sea capaz de seguir el vehículo.

- Punto a punto
- Curvas de Dubin [4]
- Curvas de Lissajous o similares [5]

Cada tipo de trayectoria distinta presentará sus ventajas e inconvenientes que se describirán a continuación. Se describirá además su método de implementación en el entorno MATLAB.

3.2.4.1 Punto a punto

Este tipo de curva se reduce exclusivamente al tipo de trayectoria que se requiere que siga el vehículo para ir de un punto A a un punto B. La trayectoria podría por ejemplo ser cualquier tipo de polígono como por ejemplo un cuadrado como podemos ver en la imagen:

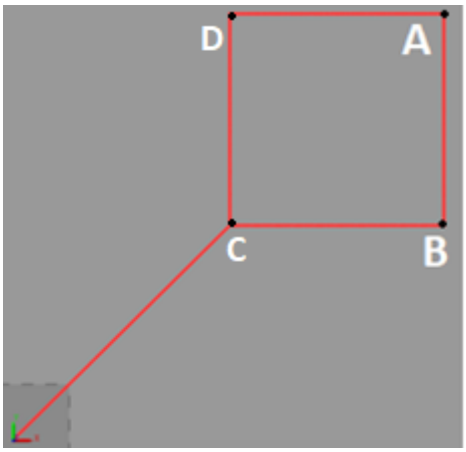


Ilustración 95: Trayectoria Punto a Punto

El objetivo del sistema de control que se pretende diseñar es que el vehículo pase por las 4 esquinas que describen el cuadrado.

El método de Pure Pursuit nos permitirá que el vehículo pueda realizar este tipo de trayectoria ya que calcula el radio de giro que requiere el vehículo para desplazarse a cualquier punto en el plano X-Y.

Conviene mencionar que este método presenta un pequeño problema a la hora de generar la trayectoria necesaria para un punto que se encuentre detrás del vehículo.

Supongamos que el vehículo se encuentra desplazándose desde el punto B de la imagen anterior, al punto A, y al alcanzar dicho punto A, quiere volver al punto B. La situación que se presentaría sería la de la siguiente imagen:

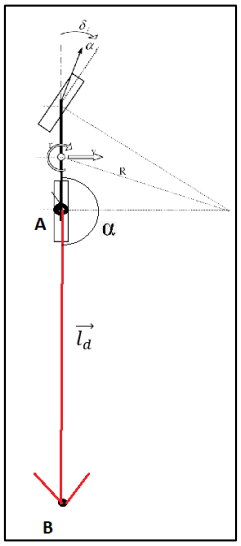


Ilustración 96: Problemática Pure Pursuit

Como puede apreciarse, el ángulo que se produce entre $\overrightarrow{l_d}$ y el vehículo es de 180º, esto al introducirlo en la fórmula que calcula el ángulo de giro de la rueda directriz nos da el siguiente resultado:

$$\delta = \arctan\left(\frac{2 \cdot L \cdot \sin(180)}{l_d}\right) = \arctan(0) = 0º$$

Es decir, tendríamos un ángulo de giro nulo, con lo que el vehículo teóricamente nunca sería capaz de alcanzar el punto B. En la propia simulación podrá observarse, en parte debido a que el mecanismo diseñado no coincide perfectamente con el modelo de la bicicleta, que el vehículo gira levemente al alcanzar el punto A y termina alcanzando el punto B pero tras recorrer una distancia excesivamente mayor a la necesaria.

Este problema se solventa en el sistema de control que se detallará en el apartado de Sistema de Control de dirección.

Un detalle importante que cabe resaltar es el hecho de que cualquier otra curva que se quiera diseñar, estará formada por cientos de tramos punto a punto, ya que nuestro sistema de control tratará de seguir una línea de puntos que forme la trayectoria deseada. En la siguiente imagen puede apreciarse el aspecto de una trayectoria formada por la curva de Dubin. Esto se explicará con mayor detalle en el apartado de diseño de controlador.

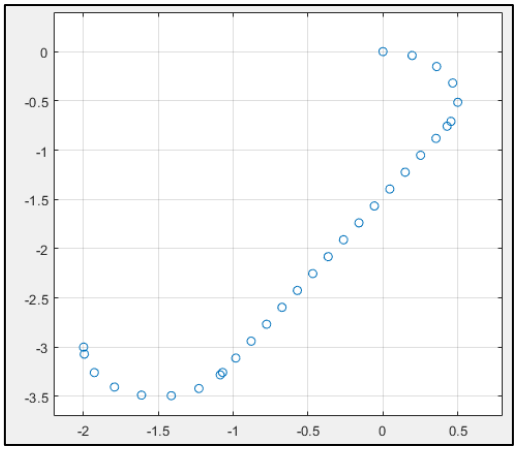


Ilustración 97: Trayectoria de puntos

Para la trayectoria punto a punto y las curvas de Dubin, usaremos el siguiente polígono irregular para comprobar la efectividad de nuestro sistema de control:

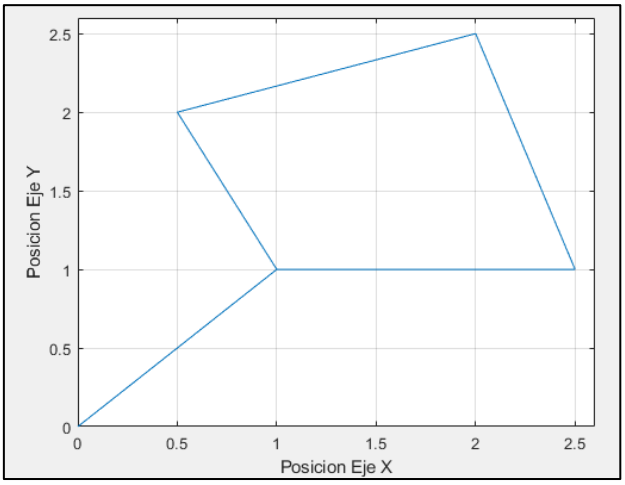


Ilustración 98:Polígono trayectoria

El motivo de que se haya escogido dicho polígono se debe a que resulta una trayectoria con varias curvas y rectas que permiten probar el sistema de control exhaustivamente.

El objetivo de la prueba de simulación es que el vehículo sea capaz de pasar por todas las esquinas que dispone el polígono. Con ello el vehículo debería pasar por las siguientes coordenadas:

$$(0,0) \rightarrow (1,1) \rightarrow (2.5,1) \rightarrow (2,2.5) \rightarrow (0.5,2.5) \rightarrow (1,1)$$

El camino que escoja el vehículo no será el mismo para una trayectoria punto a punto que usando las curvas de Dubin.

3.2.4.2 Curvas de Dubin

Las curvas de Dubin, son un tipo de curvas que tratan de encontrar el camino más corto para que un vehículo se desplace desde una posición y configuración 'A' a una posición y configuración B. Para ello se construye una trayectoria compuesta por 3 segmentos distintos que pueden ser giros con un radio *r*, que suele denominarse por la letra C, o un desplazamiento en línea recta, que suele denominarse por la letra S. Denotaremos los distintos segmentos de la siguiente manera:

- Giro a la izquierda con un radio *r* - L
- Giro a la derecha con radio *r* - R
- Tramo recto - S

Con estos 3 segmentos, Dubin creo un total de 6 trayectorias distintas: LRL, RLR, LSL, RSR, LSR, RSL. En las siguientes imágenes podemos ver algunos ejemplos en los que se aprecian como quedarían algunas de estas trayectorias dichas trayectorias:

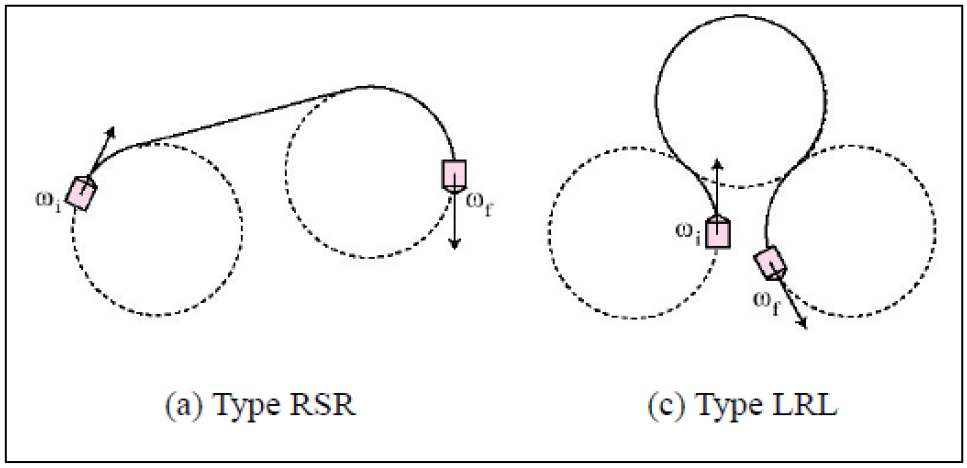


Ilustración 99: Curvas de Dubin

Este tipo de trayectorias es bastante útil en el diseño de trayectorias para un vehículo de Ackermann ya que toma en consideración el radio de giro mínimo que es capaz de realizar el propio vehículo.

Para el vehículo diseñado se ha considerado un radio de giro mínimo de 0.5 m. ligeramente mayor al valor mínimo de 0.36 m que el vehículo debería de tener en teoría.

Generación trayectoria de Dubin

Para generar la trayectoria de Dubin, nos aprovecharemos del objeto *dubinsConnection* del que dispone Matlab, que permite generar automáticamente una trayectoria de Dubin entre 2 puntos. También se requiere definir unas orientaciones al iniciar el recorrido y tras finalizarlo:

```
% Coordenadas por las que tiene que pasar el vehiculo
x = [0 1 2.5 2 0.5 1];
y = [0 1 1 2.5 2 1];

% El angulo inicial conviene que sea 0
a = [0 0 pi/2 pi -pi/2 -pi/2];
```

Ilustración 100: Condiciones trayectoria Dubin

Para cada coordenada se define un ángulo, que corresponde con el ángulo que deberá tener el vehículo al alcanzar dicho punto.

A continuación, crearemos el objeto *dubinsConnection* y definiremos el radio de giro mínimo que permite nuestro vehículo:

```
% Obtenemos la trayectoria de Dubin que deberá seguir el vehiculo
dubConnObj = dubinsConnection;
dubConnObj.MinTurningRadius = 0.5;
```

Ilustración 101: Definición dubinsConnection

A continuación, podremos proceder a crear las propias trayectorias que debe seguir el vehículo:

```
% numero de trayectos que se van a programar
n = size(x,2) - 1
```

```
% Obtenemos los 5 trayectos que deberá realizar el vehiculo
for i = 1:n
    startpose = [x(i) y(i) a(i)];
    goalpose = [x(i+1) y(i+1) a(i+1)];
    [pathSegObj, pathcosts] = connect(dubConnObj, startpose, goalpose);
    length = pathSegObj{1}.Length;
    poses = interpolate(pathSegObj{1},0:0.01:length);
    xn = poses(:,1)
    yn = poses(:,2)
    % Para evitar errores en la visualizacion en el Mechanics explorer
    if not(i == 1)
        xn(1) = [];
        yn(1) = [];
    end
    xc = vertcat(xc,xn);
    yc = vertcat(yc,yn);
end

np = size(xc,1);
```

Ilustración 102: Generación trayectoria Dubin

Ya que queremos que nuestro vehículo pase por un total de 6 puntos, necesitaremos generar un total de 5 trayectorias, para ello se ha creado un bucle *for* que genera dichas trayectorias una detrás de otra.

Primeramente, se obtienen los vectores [xi yi thetai] y [xf yf thetaf] que definirán los puntos inicial y final de cada trayectoria, así como sus orientaciones. Seguidamente se crean las trayectorias y se guardan en las variables *xc* e *yc* que usaremos en el propio modelo de Simulink para realizar el control de trayectoria. Al concatenar una trayectoria con la siguiente, se elimina el primer punto de la trayectoria, ya que de no hacerlo, no podríamos representar la trayectoria del vehículo en MechanicsExplorer.

Se crea una variable *np* con el número de puntos que componen la trayectoria. Esto nos ayudará a finalizar la simulación una vez el vehículo haya completado el recorrido programado.

Con todo esto, la trayectoria generada se puede apreciar en la siguiente imagen:

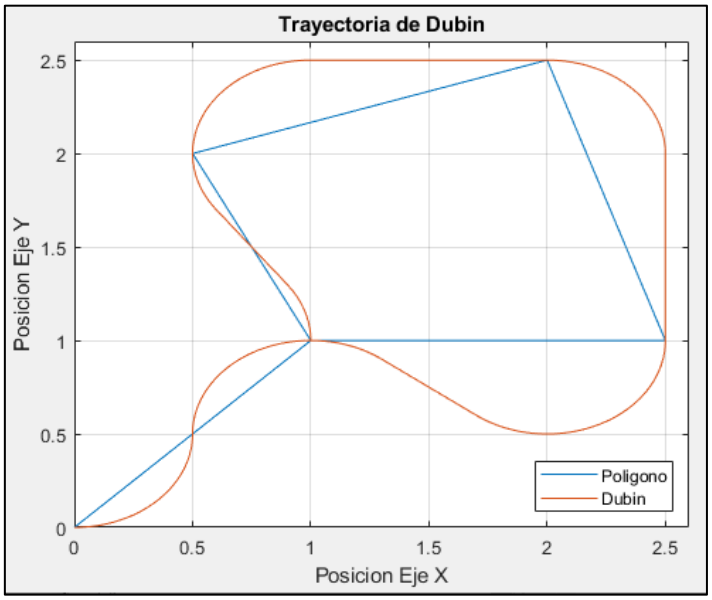


Ilustración 103: Trayectoria de Dubin

$$\frac{\omega_x}{\omega_y} = \frac{2}{3}$$

3.2.4.3 Curvas de Lissajous o similares

Otro tipo de trayectorias que suelen ser comunes en los trabajos que involucren un dispositivo que se desplaza por el plano, son las curvas de Lissajous o Lemniscata. Estas curvas no tienen como propósito alcanzar un punto concreto en el espacio, sino simplemente comprobar si el vehículo diseñado es capaz de realizar dicha trayectoria

Curva de Lissajous

La curva de Lissajous, también conocida como figuras de Lissajous, son un tipo de curvas compuestas por 2 movimientos armónicos simples cuyas direcciones son perpendiculares. Para representar dicha curva en el plano X-Y usaremos las siguientes fórmulas:

$$x = A_x \cdot \sin(\omega_x \cdot t) \qquad y = A_y \cdot \sin(\omega_x \cdot t + \delta)$$

Siendo las variables que definen la curva las amplitudes A_x y A_y, las velocidades angulares ω_x y ω_y además de δ, que representa el desfase entre las distintas ondas. Estas curvas se usan para formar distintas figuras variando la relación ω_x/ω_y y el ángulo de desfase δ.

A continuación, veremos algunos ejemplos de las figuras que se obtienen al variar los parámetros descritos anteriormente.

Figuras de Lissajous obtenidas para una relación $\omega_x/\omega_y = 1/2$:

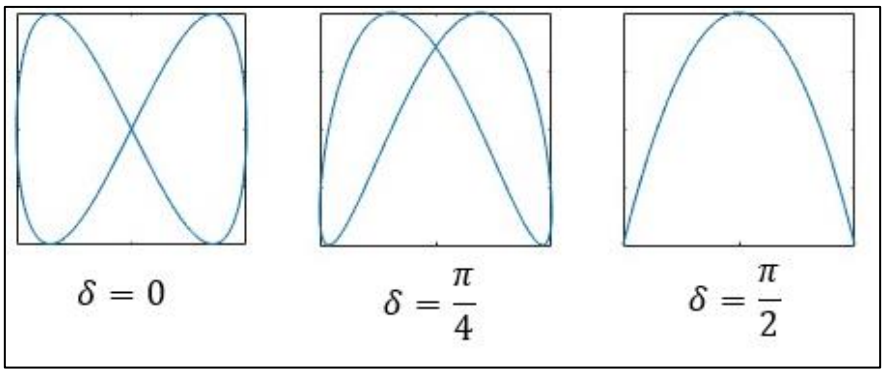


Ilustración 104: Lissajous 1/2

Figuras de Lissajous obtenidas para una relación $\omega_x/\omega_y = 2/3$:

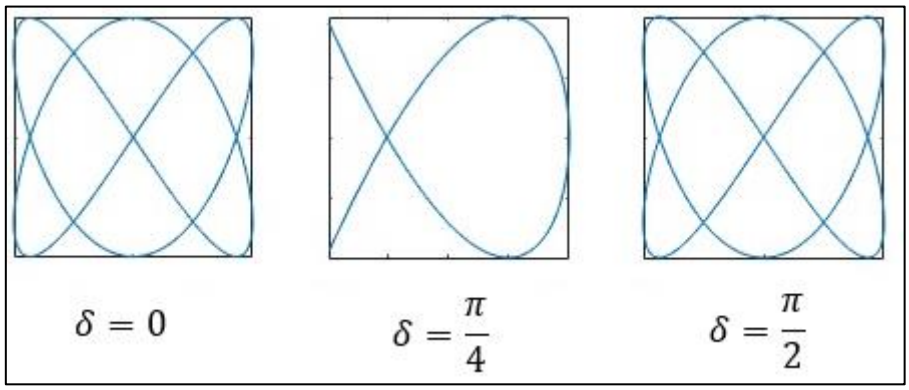


Ilustración 105: Lissajous 2/3

De estas figuras se puede comprobar que muchas de ellas contienen giros pronunciados que podrían dificultar e incluso imposibilitar un seguimiento de la trayectoria adecuado. Para ello se seleccionarán aquellas que no dispongan de giros muy bruscos o en su defecto, se procederá a aumentar su tamaño, creando con ello unas curvas con mayor radio de giro.

Las curvas que usaremos serán las siguientes:

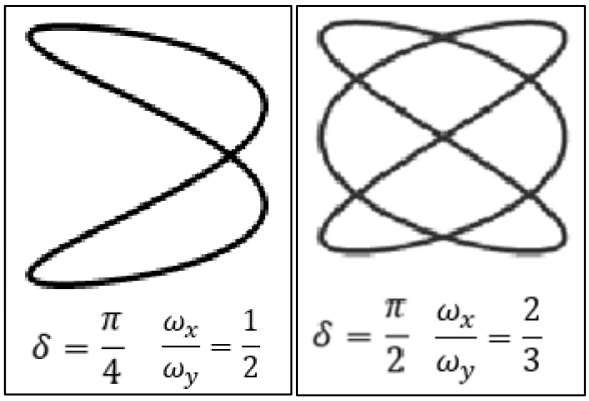


Ilustración 106: Figuras de Lissajous seleccionadas

Generación Trayectoria de Lissajous

La generación de este tipo de trayectorias sigue el mismo concepto que la generación de la Curva de Lemniscata. Se crea un array *t* con valores que van desde 0 a 2π y se le aplican las funciones definidas anteriormente en el apartado 4.2.3 a dicho array.

```
t = [0:0.01:2*pi];
wx = pi;
x = 10*sin(wx*t);

wy = 2*pi;
y = 10*sin(wy*t+pi/4);
y = y-1.4142

% Cambiamos el orden de x-y para que coincida con la imagen del Word
xc = y';
yc = x';

% Numero de puntos que compone la trayectoria
np = size(xc,1);
```

Ilustración 107: Generación Curva Lissajous 1

En el código anterior vemos como se genera la primera curva de Lissajous. Se definen unas variables ω_x y ω_y que usaremos para generar la curva de Lissajous deseada. Seguidamente solo queda aplicar las formulas que describen la curva a la variable *t* y obtendríamos las siguientes trayectorias:

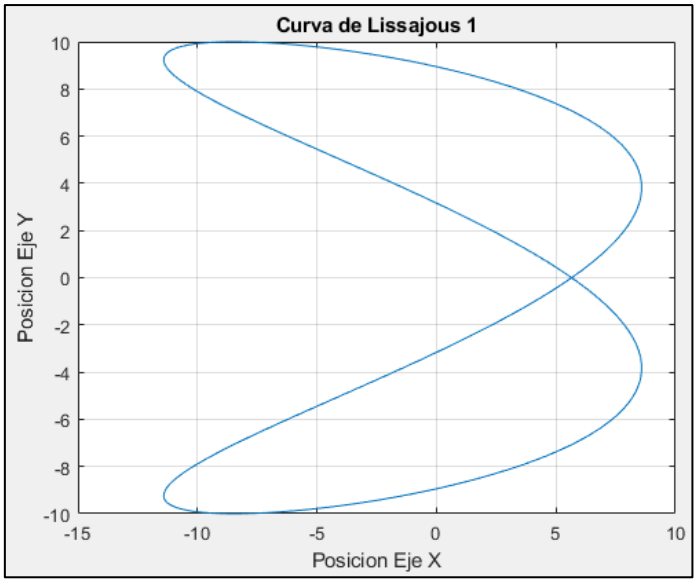


Ilustración 108: Trayectoria de Lissajous 1

Para generar la otra trayectoria que se mencionó en el apartado 4.3.2, basta con cambiar los valores de ω_x y ω_y de forma que su relación sea igual a 2/3. Además del cambio anterior, se requiere cambiar el desfase δ de $\pi/4$ a $\pi/2$. Con ello obtendríamos la siguiente curva:

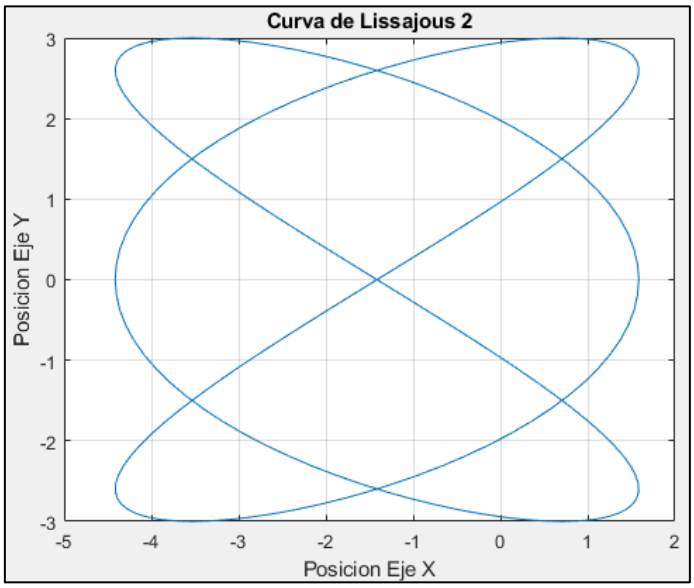


Ilustración 109: Trayectoria de Lissajous 2

Lemniscata de Bernoulli

Este tipo de curva, conocida por representar el símbolo de infinito ∞, trata de una curva en forma 8 lateral como se puede apreciar en la siguiente imagen.

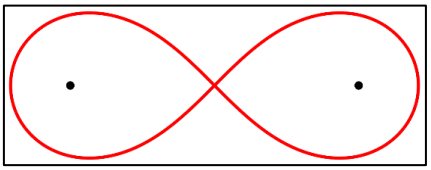


Ilustración 110: Lemniscata de Bernoulli

Para formar esta figura, se toman 2 focos, que denominaremos F1 y F2, las cuales estarán separadas por una distancia 2d. La curva se forma tomando en consideración que el producto de la distancia de ambos focos a un punto, siempre tenga por valor d^2.[6]

De esta definición se obtiene la siguiente fórmula que se puede observar gráficamente en la siguiente Ilustración:

$$\left|\overrightarrow{PF_1}\right| \cdot \left|\overrightarrow{PF_2}\right| = d^2$$

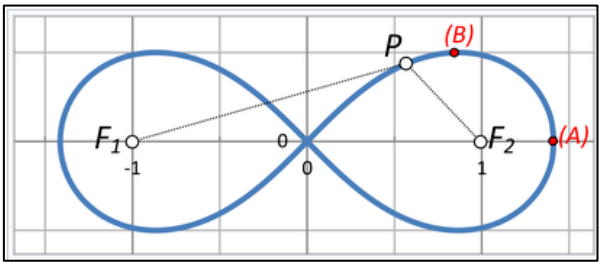


Ilustración 111: Obtención Lemniscata

Generación trayectoria de Lemniscata

Para generar estas trayectorias, nos aprovecharemos de la ecuación paramétrica que define la curva de Lemniscata:

$$x = \frac{d\sqrt{2}\cos(t)}{\sin^2(t)+1}; \qquad y = \frac{d\sqrt{2}\cos(t)\sin(t)}{\sin^2(t)+1}$$

Como se comentó anteriormente, la variable *d* representa la mitad de la distancia entre los dos focos que definen este tipo de trayectoria. La variable *t* es el ángulo sobre el que queremos realizar la figura. Ya que nos interesa obtener la figura completa, este valor irá desde 0 a 2π, es decir una circunferencia completa.

Para obtener los array *xc* e *yc* que contienen la trayectoria completa que queremos que siga el vehículo, se han creado las siguientes funciones:

```
function a = Lemnix(c,d)

    a = d * sqrt(2) * cos(c)/((sin(c))^2 + 1);
end

function a = Lemniy(c,d)

    a = d * sqrt(2) * cos(c)*sin(c)/((sin(c))^2 + 1);
end
```

Ilustración 112: Funciones Lemniscata

A estas funciones le pasaremos como argumento el vector *theta* que contiene un barrido de ángulos desde 0 a 2π.

```
d = 1;
theta = [0:0.01:2*pi]';

xc = arrayfun(@(x) Lemnix(x,d),theta);
yc = arrayfun(@(x) Lemniy(x,d),theta);
```

Ilustración 113: Generación Lemniscata

Mediante la función arrayfun(), se han aplicado las funciones Lemnix() y Lemniy() a todos los ángulos que deseábamos, obteniendo con ello la curva de Lemniscata como podemos observar en la imagen inferior:

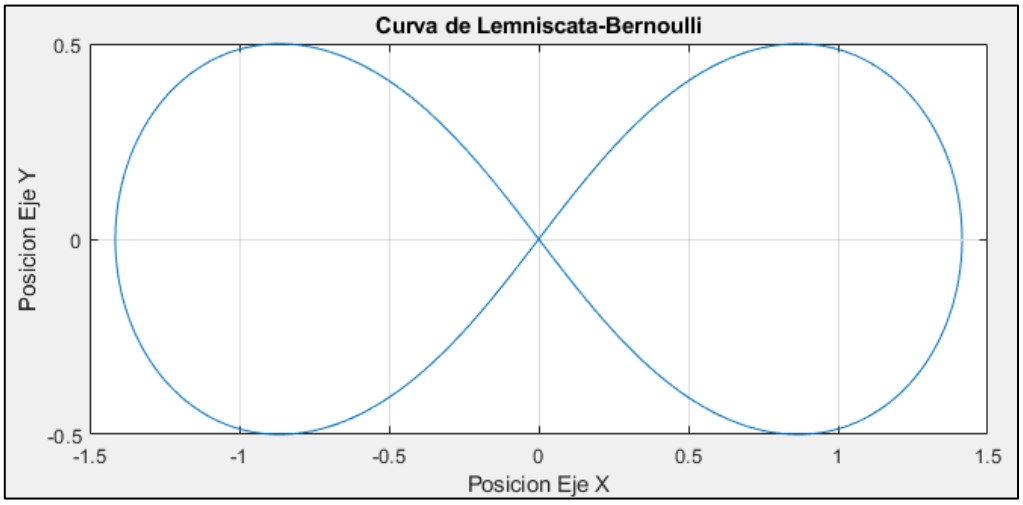


Ilustración 114: Trayectoria de Lemniscata

Nota: Todas las trayectorias que se han definido anteriormente tienen un punto inicial y orientación distintas, de forma que se dificultaría en gran medida realizar una prueba de seguimiento de trayectoria para una misma condición inicial. Para solucionar esto, cada script que genera la trayectoria incluye las siguientes líneas que colocan correctamente el vehículo al iniciar la simulación.

```
20      % Le damos un angulo inicial al vehículo para alinearlo con la trayectoria
21 -    thetai = atan2((yc(2)-yc(1)),(xc(2)-xc(1)));
22
23      % Posicion inicial del vehiculo
24 -    x0 = xc(1);
25 -    y0 = yc(1);
```

Ilustración 115: Compensación Inicial

3.2.5 Sistema de Control de dirección

A continuación, describiremos como se ha implementado en Simulink el sistema de control de dirección diseñado para el vehículo de Ackermann.

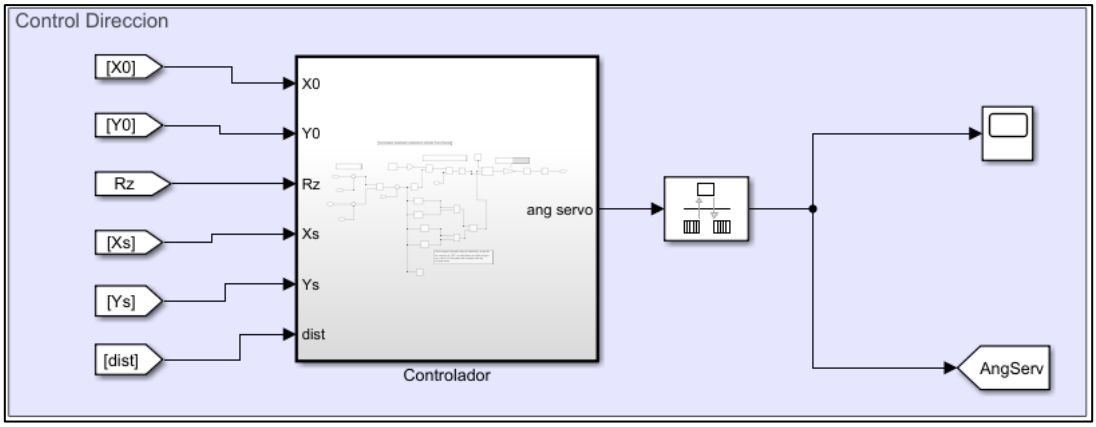


Ilustración 116 Control de dirección

Primeramente recordaremos las ecuaciones introducidas en capítulos anteriores que nos permiten obtener el ángulo de giro necesario para seguir la trayectoria predefinida.

Del capítulo 2 recordamos que el método utilizado para lograr un sistema de control capaz de lograr que un vehículo de Ackerman siga una trayectoria es el método de Pure Pursuit. En la siguiente imagen pueden apreciarse las distintas variables que intervienen en dicho sistema de control:

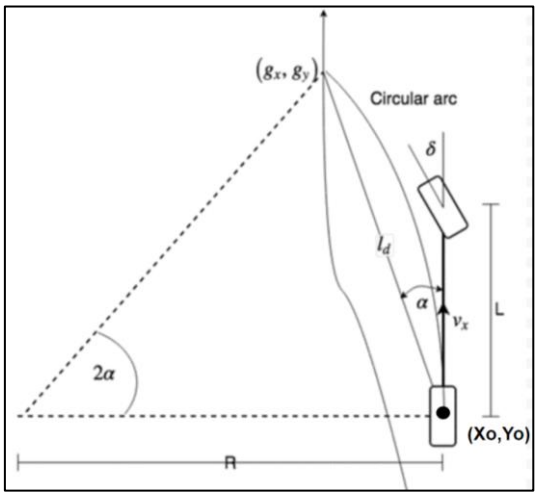


Ilustración 117: Control Pure Pursuit

El sistema de control genera un radio de giro de forma que el eje trasero pase por el punto que se trata de alcanzar. Esto se logra dándole a las ruedas un ángulo de giro definido por la siguiente fórmula:

$$\delta = \arctan\left(\frac{2 \cdot L \cdot \sin(\alpha)}{l_d}\right)$$

Vemos que las únicas variables que influyen en el ángulo de giro requerido son la distancia del centro del eje trasero al punto objetivo (l_d), variable que ya se obtuvo en el subconjunto anterior, y el ángulo que forma el vehículo con una línea recta que une dichos puntos (α), ya que la variable L es una constante que representa la distancia entre ejes del vehículo. A continuación, veremos como se ha obtenido dicho ángulo.

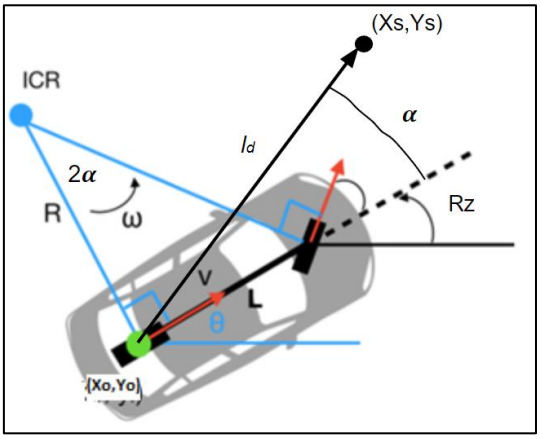


Ilustración 118: Cálculo alpha

En la imagen anterior se aprecian todas las variables necesarias para obtener el ángulo α mediante unos sencillos cálculos trigonométricos. El dibujo anterior se puede simplificar de la siguiente manera:

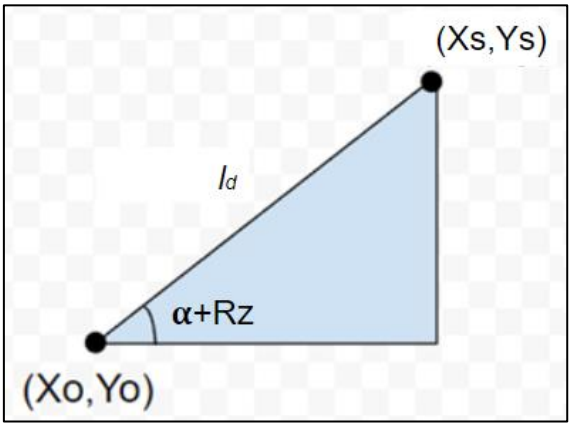


Ilustración 119: Triángulo alpha

Vemos que para calcular el ángulo α, basta con obtener el ángulo que produciría un triángulo rectángulo entre la posición del vehículo y la posición objetivo. Con ello la formula a resolver se reduciría a:

$$\alpha = \text{atan}\left(\frac{Ys - Yo}{Xs - Xo}\right) - Rz$$

Si utilizásemos la función arcotangente normal, el sistema de control no funcionaría para los casos en los que el ángulo (α+Rz) fuese mayor que π/2 o menor que -π/2, ya que la función arcotangente solo es capaz de devolver valores que se encuentren en el primer y cuarto cuadrante, es decir que se encuentren entre -π/2 y π/2.

Debido a ello en la implementación en Simulink nos aprovecharemos de la función atan2() de Matlab, la cual es capaz de devolver valores de los cuatro cuadrantes, es decir valores en el rango de -π a π.

Como ya se mencionó en el apartado anterior, al encontrarse el punto que se pretende alcanzar justo detrás del vehículo, puede ocurrir que el valor del Radio de giro obtenido sea excesivamente elevado, llevando a un ángulo de giro reducido que tardaría un tiempo excesivo en alcanzar el punto deseado.

Para resolver este problema se ha incluido en el diagrama de Simulink un conjunto de bloques lógicos que para el caso descrito, sustituyen al sistema de control automático, provocando una señal de giro coincidente con el máximo ángulo de giro permitido por el vehículo. De esta forma el vehículo realizará un giro brusco hasta lograr alinearse mejor con el punto objetivo de la trayectoria.

A continuación, se puede apreciar el diagrama de Simulink que implementa los cálculos descritos anteriormente:

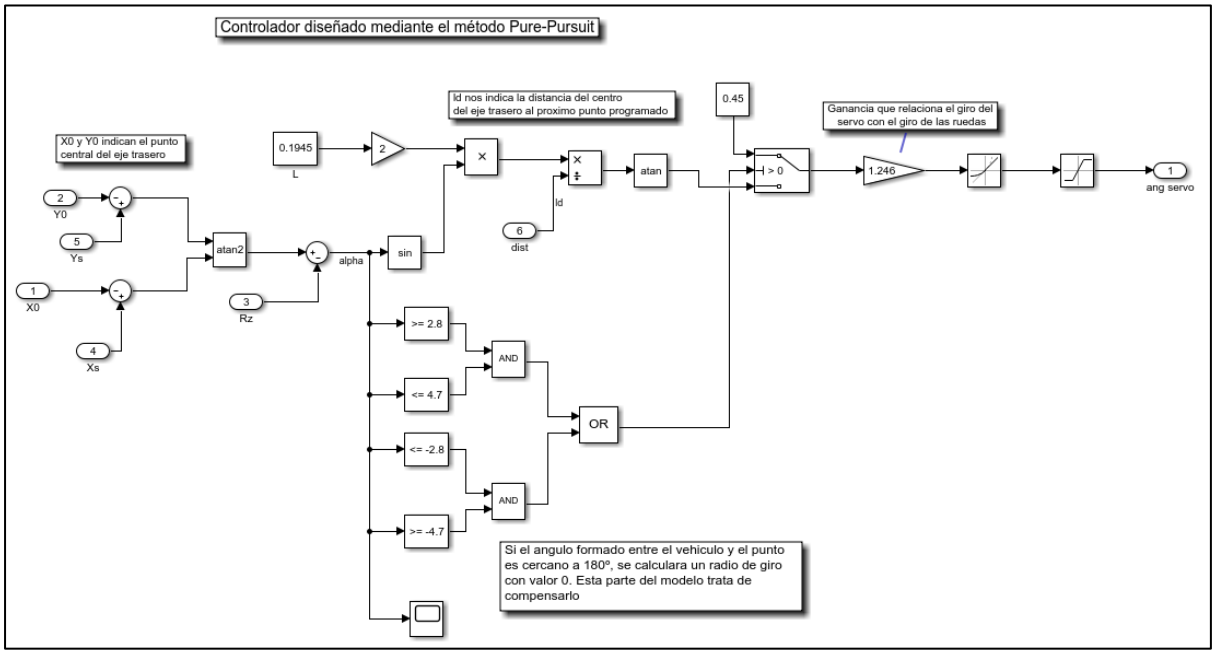


Ilustración 120: Diagrama Sistema de Control

El diagrama comienza en el lado izquierdo obteniendo el valor del ángulo α implementando la fórmula que se describió anteriormente.

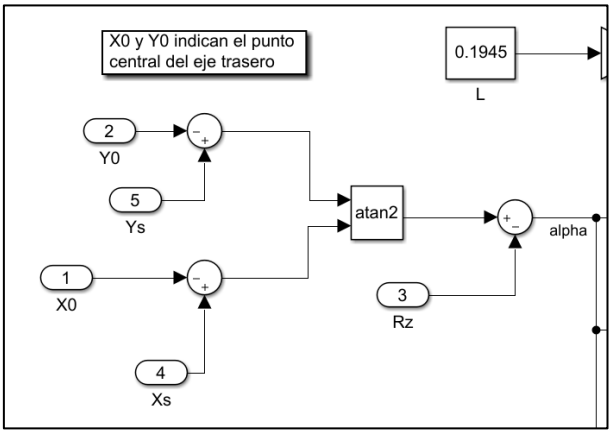


Ilustración 121: Cálculo alpha

Seguidamente a partir de dicho valor, se procede a calcular el ángulo de giro de las ruedas directrices, o en el caso de que el punto se encuentre detrás del vehículo, a aportarle una señal constante a la señal de giro. En la siguiente imagen puede apreciarse el diagrama diseñado para este último caso:

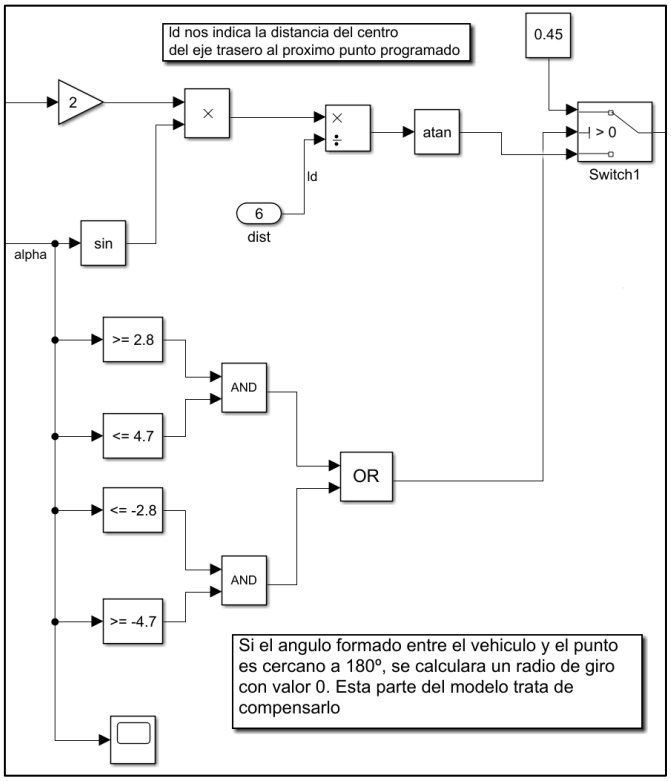


Ilustración 122: Modelo lógico

Este diagrama lógico, producirá un valor de 1, en la entrada del bloque 'Switch1', si el valor del ángulo α calculado se encuentra cercano al valor de π o $-\pi$. Con esto le proporcionaríamos a la rueda un ángulo de giro de 0.45 radianes, correspondiente a unos 25º.

Finalmente, en la siguiente imagen puede apreciarse como el diagrama de Simulink obtiene finalmente el ángulo de giro que le debemos proporcionar al servomotor para que el vehículo siga la trayectoria definida:

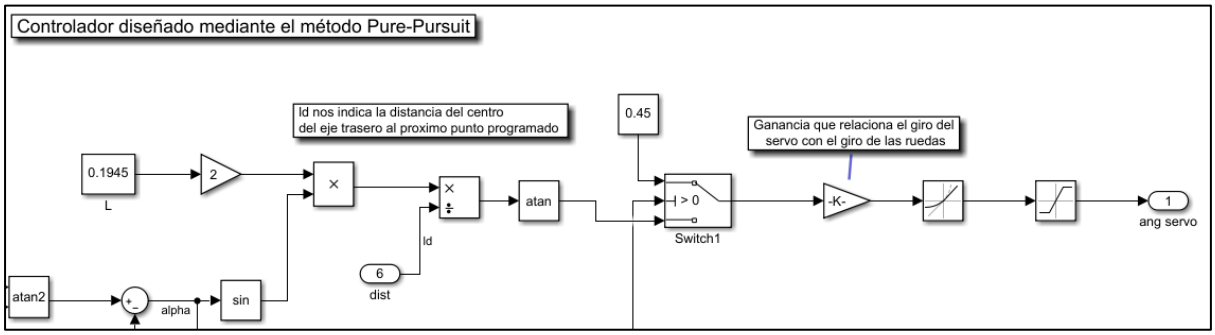


Ilustración 123: Diagrama Pure Pursuit

Una vez calculado el ángulo de giro de las ruedas, conviene aplicar una ganancia K, para compensar esa diferencia entre el ángulo girado por el servomotor y el ángulo girado por las propias ruedas. Esta ganancia se obtendrá en función de las primeras simulaciones que se realicen sin un sistema de control. Finalmente se incluyen unos bloques de Rate Limiter, para simular el tiempo que tardaría el Servomotor en realizar el giro, y un bloque de 'Saturation' para que el ángulo que le apliquemos al servomotor no supere unos límites definidos por la propia construcción del vehículo.

3.3 Resultados Simulaciones

En el siguiente apartado se pretende describir las distintas simulaciones realizadas así como los resultados obtenidos, los cuales mostraran el comportamiento de nuestro vehículo.

Para ello se comenzará el capítulo sin ningún tipo de controlador, para poder observar claramente como se mueve el vehículo en el espacio. Seguidamente incorporaremos el controlador PI de velocidad de forma que podamos controlar adecuadamente la velocidad del vehículo. Finalmente entraremos en la simulación del sistema de seguimiento de trayectoria diseñado para el vehículo.

A la hora de realizar el control del vehículo con el controlador de velocidad/trayectoria, tomaremos en consideración que el microcontrolador que se pretende utilizar en el montaje real es capaz de funcionar con un período de 10 ms.

3.3.1 Control manual

En este primer nivel de control se pretende lograr definir el comportamiento más básico del vehículo sin entrar en detalles del control del mismo. En este tipo de simulaciones se trata de modificar las variables aplicadas a las 2 entradas del sistema (par del eje y dirección del servomotor) y observar su comportamiento.

Para ello se van a realizar las siguientes simulaciones:

- Par fijo y dirección nula
- Par con aumento gradual y dirección nula
- Par fijo y giro derecha/ izquierda
- Par variable y giro variable

En cada una de las simulaciones mencionadas se tratará de estudiar un aspecto del comportamiento del vehículo.

Par fijo y dirección nula

En esta simulación se le aplicarán distintos valores fijos al par aplicado al eje, manteniendo fijo el servomotor que controla la dirección del vehículo. El objetivo es observar el comportamiento del vehículo al aplicarle un par en el eje similar al que se le aplicaría al montaje real. Se tomará en consideración que el máximo par aplicable por el motor disponible para el montaje es de 1.3Nm.

Comenzaremos la simulación aplicándole al motor un par constante de 0.4Nm sobre el eje trasero mientras mantenemos la dirección fija.

Una condición que resulta indispensable para el tipo de simulaciones que pretendemos realizar, es la condición de no deslizamiento. Para comprobar que se cumple esta condición, compararemos la velocidad real del vehículo comparada con la velocidad de desplazamiento de la rueda calculada a partir de su velocidad de giro. En caso de que no haya deslizamiento, estas variables deberían tener el mismo valor.

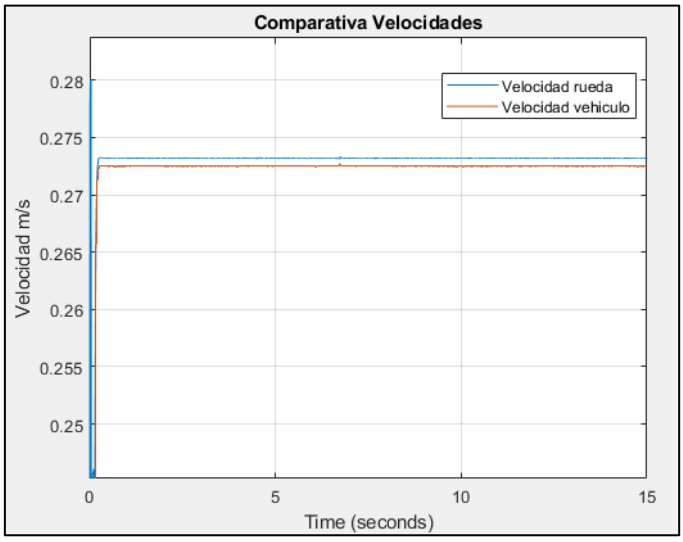


Ilustración 124: Velocidades rueda-vehículo – 0.4 Nm

De la imagen anterior se aprecia una leve diferencia entre la velocidad real del vehículo y la velocidad de la rueda calculada a partir de su velocidad de giro. Esta diferencia es lo suficientemente pequeña (del orden de 0.3% de diferencia), como para considerar que las ruedas del vehículo giran sin deslizamiento alguno.

Se aprecia además una diferencia entre ambas velocidades al inicio de la simulación. Esto se debe a que le estamos aplicando a la rueda un par instantáneo de 0.4 Nm, partiendo el vehículo de reposo, con lo que el vehículo completo no es capaz de acelerar de forma tan repentina como las ruedas del vehículo.

A continuación, realizaremos la misma prueba para el caso en el que le aplicamos al eje un par de 0.8Nm:

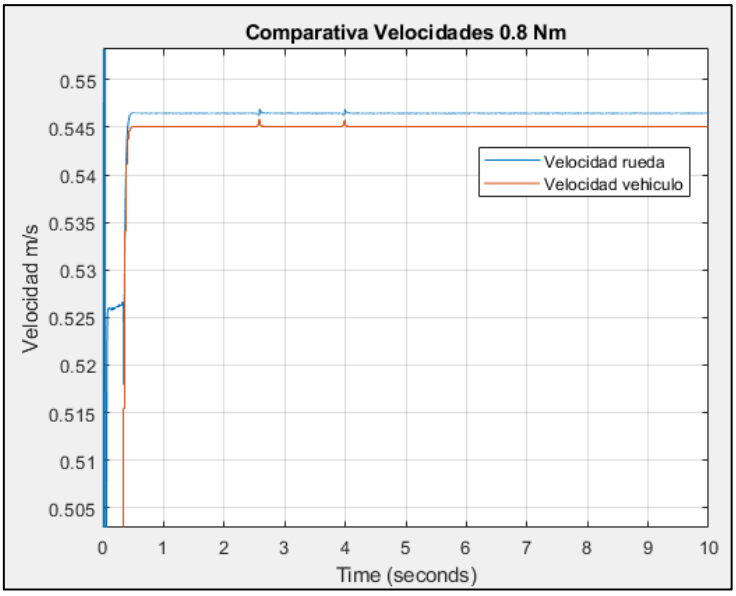


Ilustración 125: Velocidades rueda-vehículo 0.8 Nm

Igual que sucede en el caso anterior, se aprecia una leve diferencia entre ambas velocidades, del orden de 0.3%, con lo cual la diferencia sigue siendo negligente.

A continuación veremos lo que sucede si le aplicamos el par máximo de 1.3 Nm que es capaz de aportar el motor escogido.

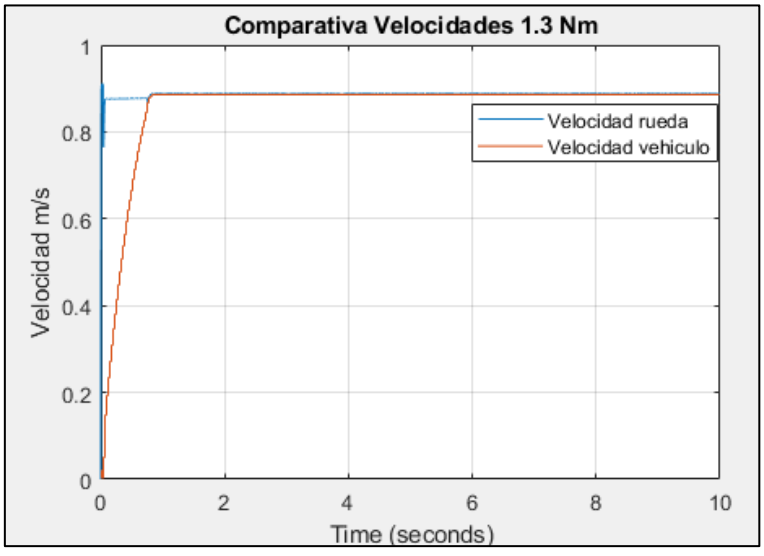


Ilustración 126: Velocidades rueda-vehículo 1.3 Nm (1)

De la gráfica anterior se aprecia como al inicio de la simulación la velocidad teórica de la rueda parte de una velocidad de 0.9 m/s mientras que la velocidad real del vehículo aumenta de forma gradual hasta alcanzar dicha velocidad. Esto se debe a que la rueda se encuentra deslizando, y por ello la fuerza de fricción entre las ruedas y el suelo se reduce, provocando con ello ese aumento en el tiempo de alcanzar la velocidad máxima.

Sin embargo si comparamos ambas velocidades en su régimen estacionario, veremos que la diferencia entre ambas velocidades se encuentra en el orden de los casos anteriores, del 0.3 % de diferencia.

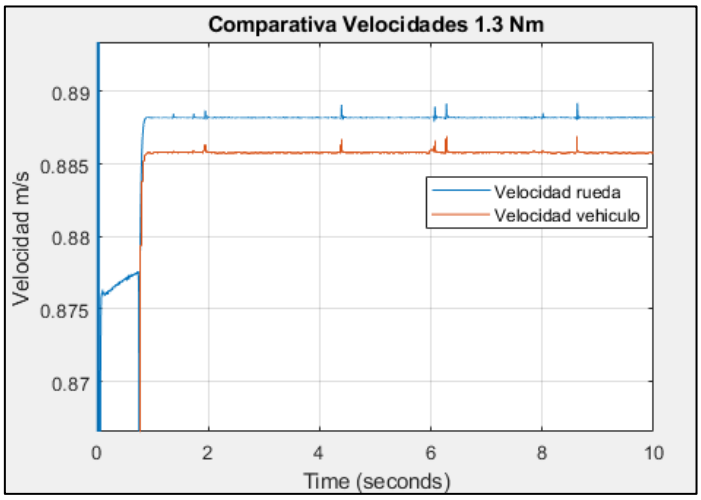


Ilustración 127:Velocidad rueda-vehículo 1.3 Nm (2)

De estas gráficas anteriores se obtiene además un estudio de la velocidad del vehículo a distintos pares en el eje, obteniendo como velocidad máxima 0.88 m/s (3.17 km/h)

Además de estudiar la velocidad del vehículo conviene estudiar su desplazamiento en el plano X-Y.

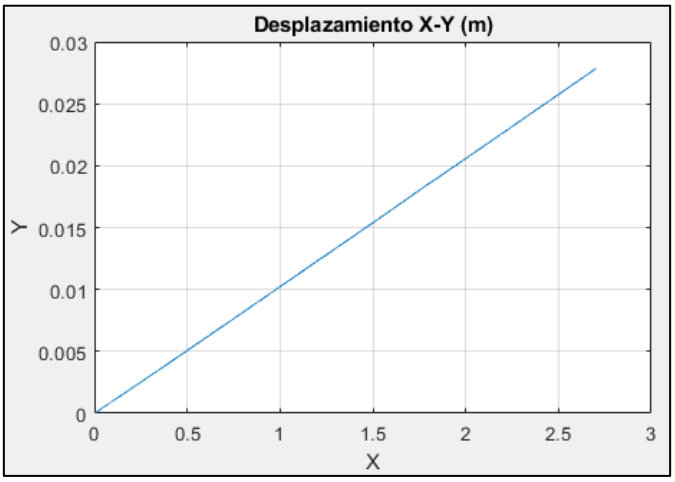


Ilustración 128: Desplazamiento X-Y

Vemos que el vehículo tiene una ligera tendencia a desplazarse hacia la izquierda. Alrededor de 1 cm hacia la izquierda por cada metro avanzado. Esto no supone un problema mayor por

no ser un desvío excesivamente elevado, por no mencionar que es un problema presente incluso en vehículos comerciales.

En los siguientes estudios que no involucren un cambio en la dirección, nos centraremos principalmente en la velocidad alcanzada del vehículo.

Par con aumento gradual y dirección nula

En esta simulación se le ha aplicado al eje del vehículo un par que aumenta de forma constante. En una primera simulación se ha aplicado una rampa con una pendiente de 0.1, logrando al final de la simulación un par aplicado más elevado del que permitiría el motor escogido.

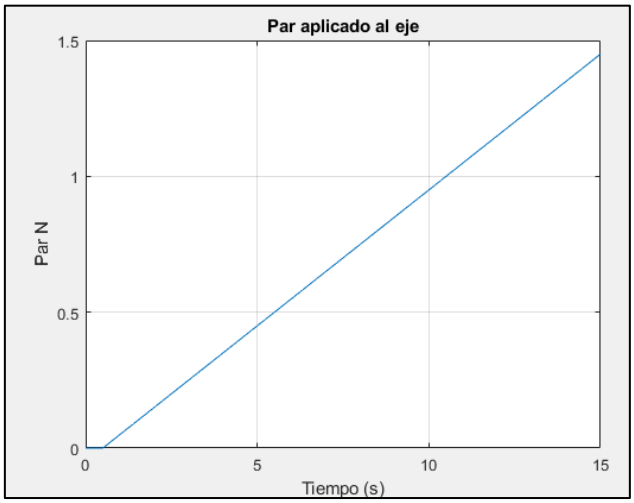


Ilustración 129: Par - rampa aplicada al eje (1)

En la imagen anterior se aprecia como aumenta el par aplicado al vehículo.

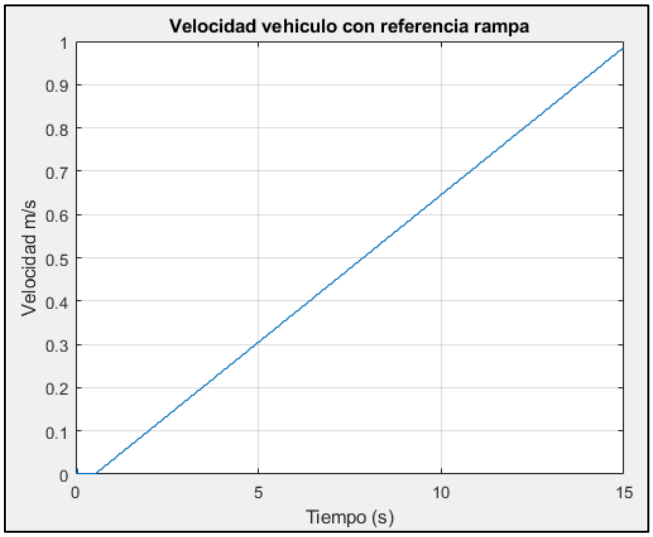


Ilustración 130: Velocidad vehículo con Referencia gradual

De la gráfica anterior se puede comprobar como la velocidad del vehículo aumenta gradualmente a lo largo de la simulación, siempre con la misma pendiente.

En una siguiente simulación, se ha decidido comprobar el funcionamiento del vehículo si se le aplica una rampa con una pendiente elevada. El objetivo de esta simulación es tratar de averiguar a partir de que velocidad comienzan a deslizar las ruedas del vehículo y que par se le aplica en ese instante.

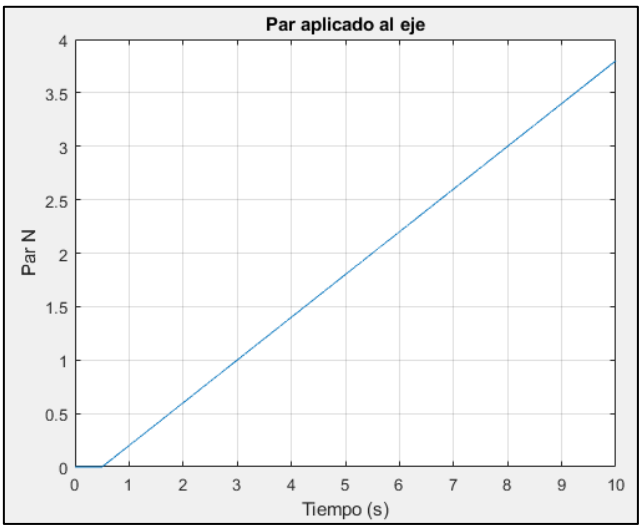


Ilustración 131: Par - rampa aplicada al eje (2)

En la imagen anterior puede apreciarse el par que le hemos aplicado al vehículo durante la simulación. A continuación veremos la velocidad del vehículo durante dicha simulación.

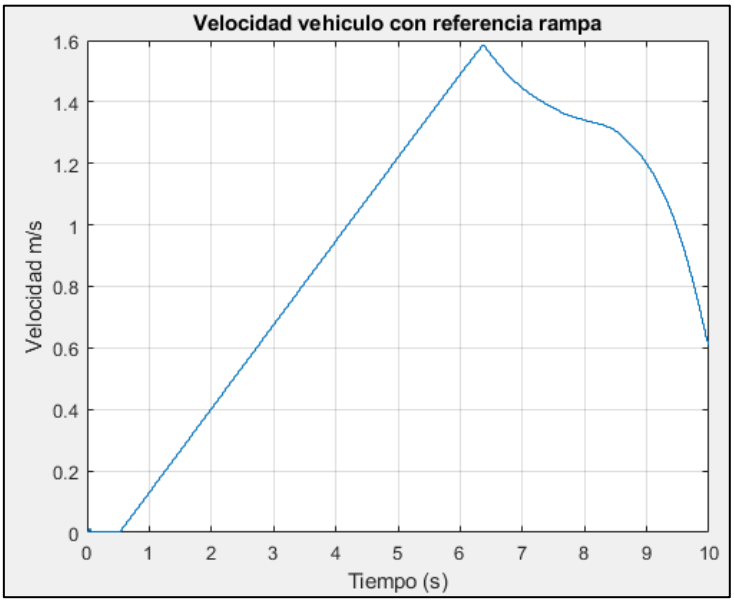


Ilustración 132:Velocidad con rampa k=0.4

En dicha gráfica se aprecia que la velocidad del vehículo comienza a reducirse en el segundo 6.4 de la simulación, momento en el que el vehículo había logrado alcanzar una velocidad de 1.58 m/s (4.74 km/h). Es en este momento en el que las ruedas del vehículo no logran mantener la fricción con el suelo y comienzan a deslizar. En el momento que comienza a deslizar, el par aplicado a las ruedas es de 2.35 Nm. Dicho deslizamiento puede observarse en la gráfica de posición del vehículo, en el que se aprecia como el vehículo pierde el control sobre su dirección.

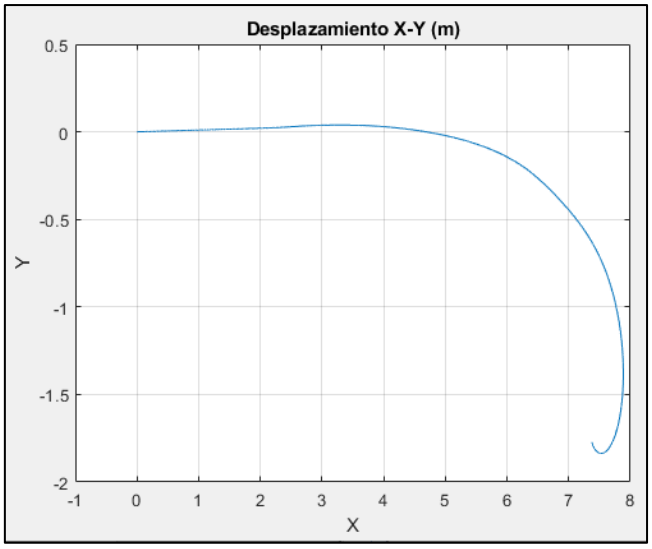


Ilustración 133: Perdida de control vehículo

De la misma manera que el vehículo deslizará si se le aplica un par en el eje demasiado elevado, también deslizará si la pendiente de dicho par aplicado resulta demasiado elevado, a continuación trataremos de obtener la pendiente máxima que podríamos aportarle al vehículo sin que comience a deslizar.

En la siguiente imagen puede verse lo que sucede si al vehículo le aplicamos un par en forma de pendiente con una rampa de pendiente m = 1:

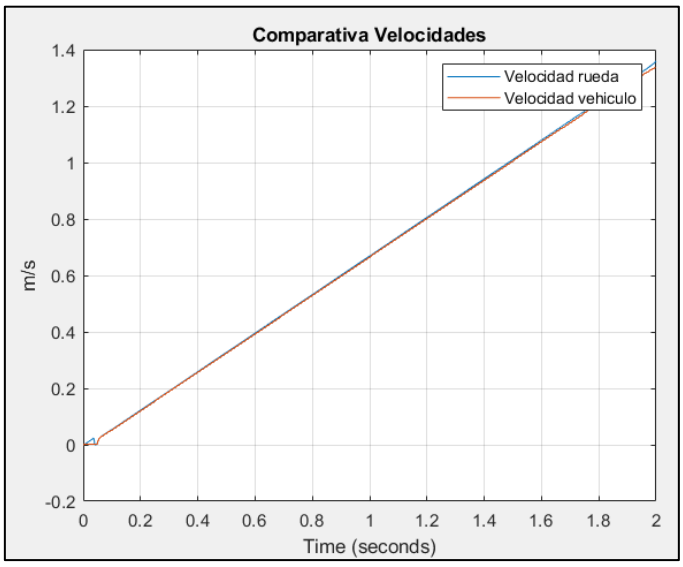


Ilustración 134: Velocidad con pendiente k = 1

Como puede verse, salvo un pequeño desvío inicial, la velocidad teórica de las ruedas del vehículo coincide durante toda la simulación con la velocidad de desplazamiento del vehículo. A continuación, veremos que sucede si subimos la pendiente de dicha señal aplicada al eje a m=4.

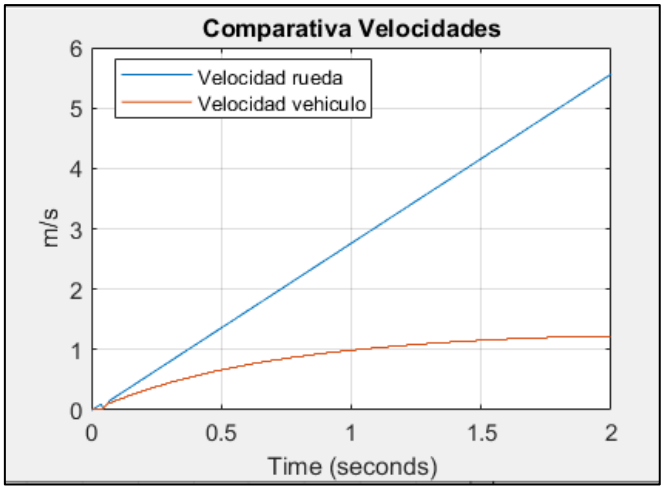


Ilustración 135: Velocidad con pendiente elevada

En esta gráfica se observa como al aplicarle al eje un par con pendiente elevada, las ruedas del vehículo deslizan y la velocidad del vehículo no coincide con la velocidad aportada por las ruedas.

Para obtener la rampa máxima que le podemos aplicar al vehículo, se han realizado varias simulaciones con distintos niveles de rampa hasta obtener la pendiente máxima que le podemos aplicar al vehículo.

Tras varias simulaciones se ha observado que la máxima pendiente de par que le podemos aplicar al eje sin que deslicen las ruedas, como puede observarse en la siguiente imagen, es de m = 3.9, la cual tiene el siguiente aspecto:

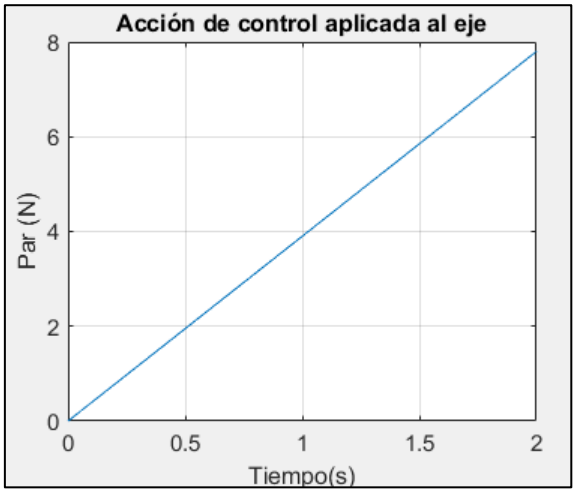


Ilustración 136: Rampa aplicada al eje

Las velocidades relativas vehículo-rueda obtenidas con dicha rampa se pueden apreciar a continuación:

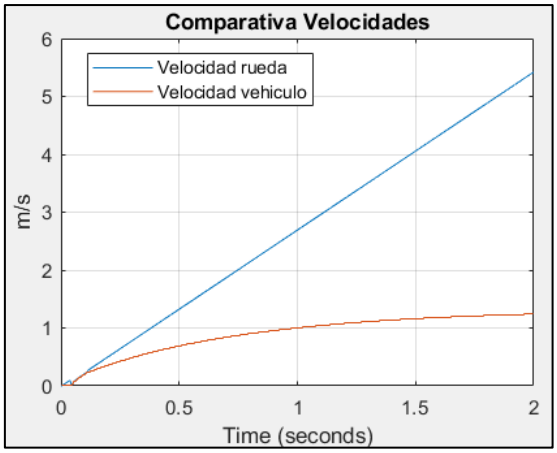


Ilustración 137: Velocidad rueda-vehículo

Como puede apreciarse, la velocidad real del vehículo y la velocidad teórica de desplazamiento de las ruedas, no coincide en ningún momento de la simulación. Con ello se concluye que m=3.9 es la pendiente máxima que se le puede aplicar al eje del vehículo.

Par fijo y giro derecha/ izquierda

A continuación, comenzaremos el estudio del giro del vehículo. Para ello le aplicaremos al vehículo un par sobre el eje fijo, mientras variamos el ángulo de giro de las ruedas del vehículo.

El objetivo de estas simulaciones es principalmente la de comprobar si el mecanismo de Ackermann diseñado cumple con su función.

Para ello realizaremos una simulación en la que le aplicamos al servomotor una señal de giro de 20º. Para evitar problemas de simulación, este valor de giro se alcanzará aplicándole al servomotor una rampa hasta alcanzar el giro deseado. En la siguiente imagen puede verse la señal de giro que le aplicamos al servomotor.

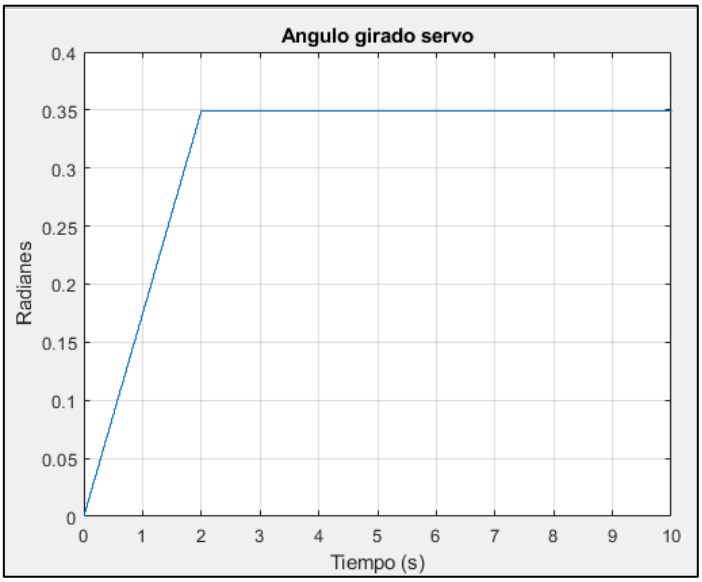


Ilustración 138: Ángulo de giro servo (1)

Este giro en el servomotor, provoca que el vehículo gire con una trayectoria que estudiaremos a continuación. En la siguiente imagen puede apreciarse el recorrido realizado por el vehículo en el plano X-Y:

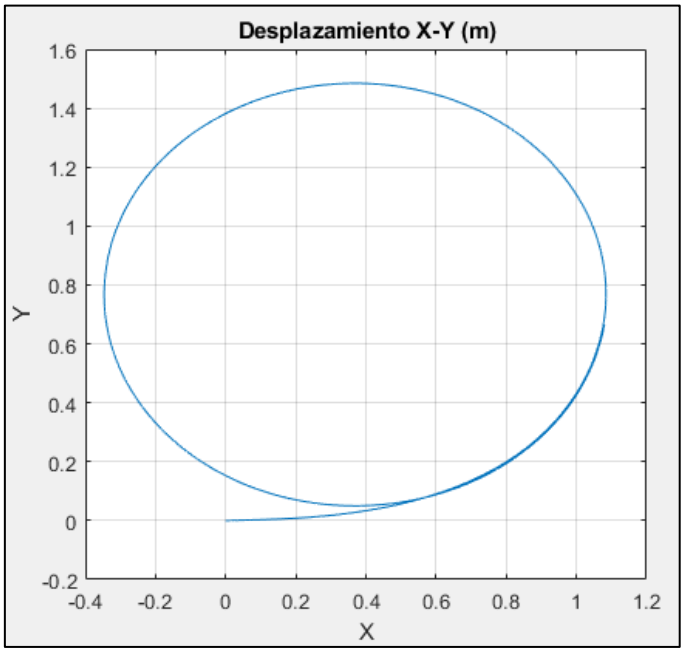


Ilustración 139: Desplazamiento X-Y giro simple

Como es de esperar, el vehículo gira hacía su izquierda realizando una trayectoria circular con un radio aproximado de 0.7 m.

Seguidamente mostraremos el ángulo girado por cada rueda así como el ángulo girado por el propio vehículo.

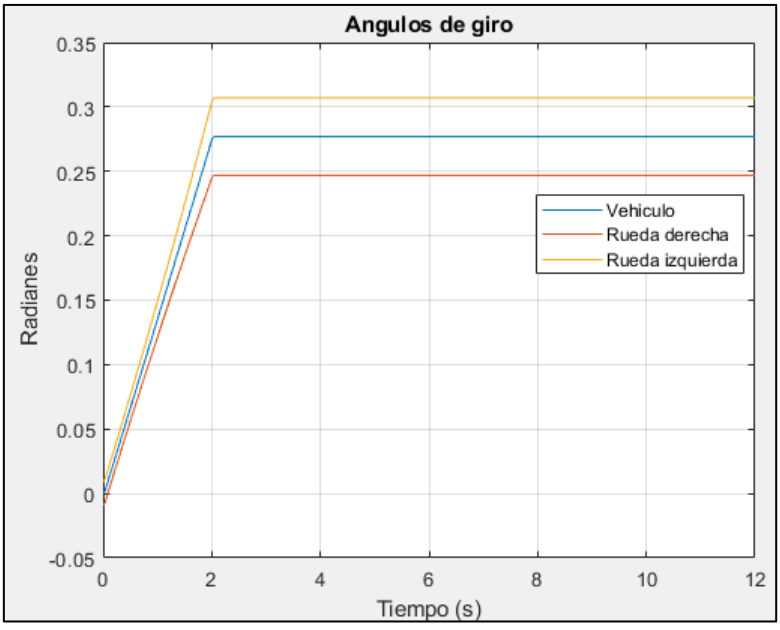


Ilustración 140: Ángulos de giro vehículo

Puede apreciarse como la rueda izquierda tiene un ángulo de giro mas elevado ya que la rueda que se encuentre en la parte interior del giro realizado siempre requerirá un mayor ángulo de giro para poder realizar la curva sin deslizamiento alguno.

A continuación, veremos si el mecanismo diseñado cumple realmente con el propósito deseado: lograr que ambas ruedas directrices formen una curva de giro con el mismo radio.

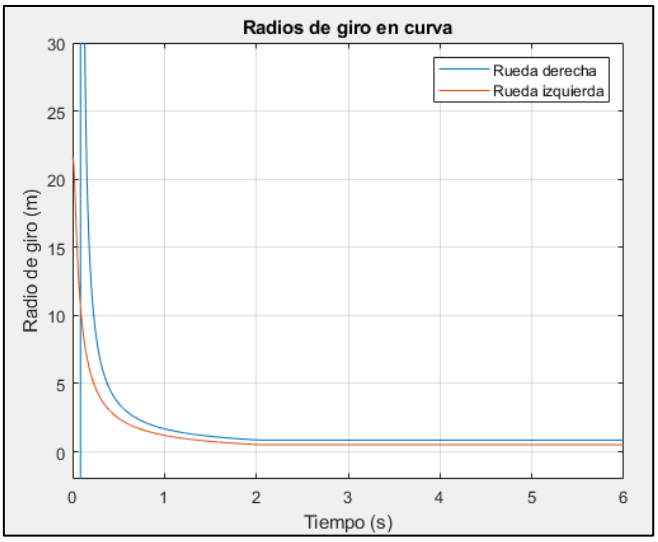


Ilustración 141: Radios de giro vehículo

De la gráfica anterior se puede apreciar como se asemejan los radios de giro obtenidos para cada rueda, con ello podemos concluir que el mecanismo diseñado cumple con el propósito diseñado, a pesar de no hacerlo a la perfección. Al inicio de la simulación tienen una desviación mayor debido al paso de la rueda derecha por un ángulo de giro nulo, causando radios de giro teóricamente infinitos.

A continuación realizaremos una nueva simulación para comprobar como se relacionan el ángulo de giro del servomotor con el ángulo de giro del vehículo. Para ello le aplicaremos al servomotor una rampa que aumente gradualmente desde los 0 hasta los 40º.

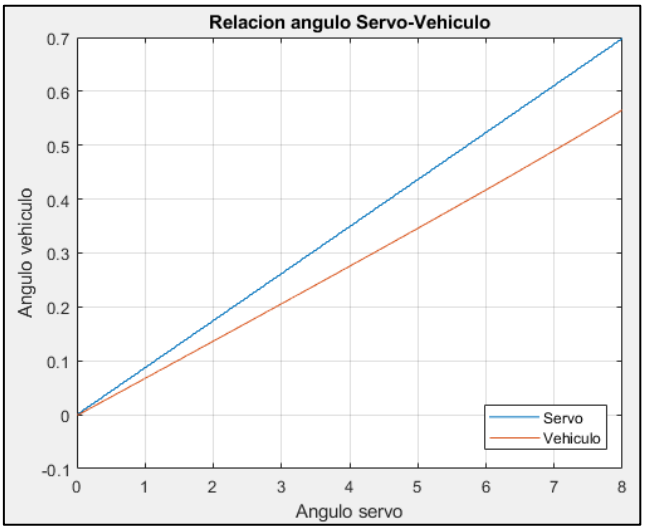


Ilustración 142: Relación ángulos de giro

Como puede apreciarse la relación entre ambos ángulos girados es lineal, cambiando únicamente el valor de su pendiente, la cual podemos obtener de forma precisa con la herramienta Curve fitting toolbox de Matlab.

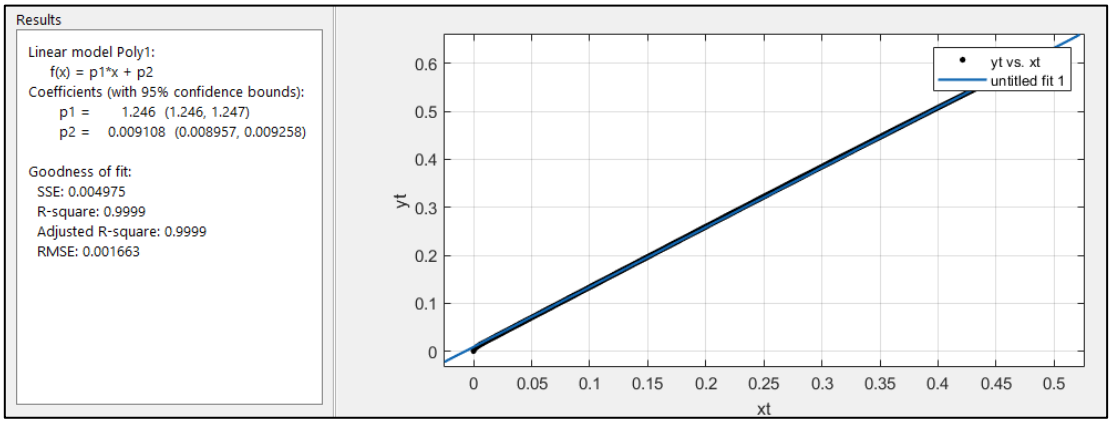


Ilustración 143: Curve Fitting Toolbox

En la imagen anterior se aprecia como se ha utilizado dicha herramienta para obtener la relación existente entre estas dos variables. De aquí se obtiene que el valor del ángulo girado por el vehículo será 1.246 veces el ángulo girado por el servo. Esta constante nos servirá además en el sistema de control de dirección en el que necesitaremos obtener un ángulo de giro de las ruedas necesario para seguir la trayectoria.

Par variable y giro variable

En la siguiente simulación que se pretende realizar, trataremos de estudiar el comportamiento del vehículo al variar el giro del vehículo durante un tramo con un par en el eje fijo y otra con la dirección fija y par en el eje variable. El objetivo de estas simulaciones es comprobar la influencia que un sistema tiene sobre el otro.

Comenzaremos estudiando la velocidad del vehículo en el caso de aplicarle un par constante de 0.8 Nm y un ángulo de giro variable sobre el servomotor.

A continuación podremos observar la acción de control que se aplica sobre el servomotor:

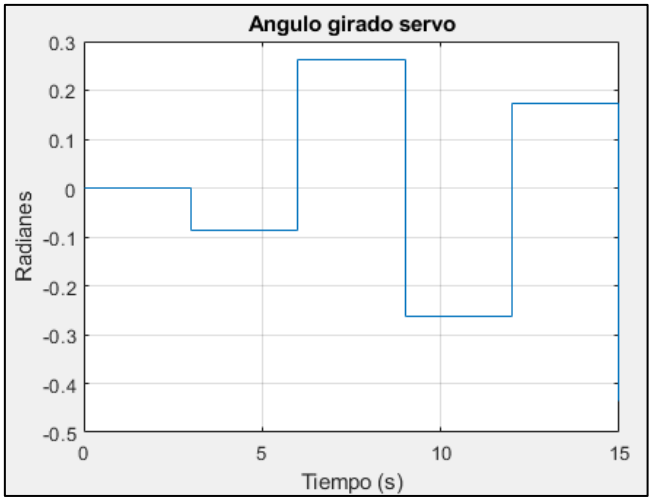


Ilustración 144: Angulo de giro servo

Dicha señal de control se compone de escalones positivos y negativos, de forma que el vehículo realice una trayectoria en zig-zag. Dicha trayectoria sobre el plano se aprecia en la siguiente imagen:

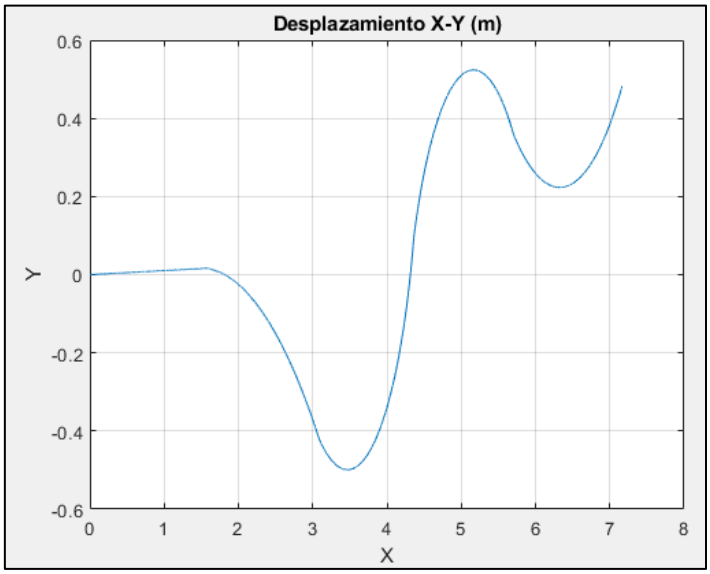


Ilustración 145: Desplazamiento X-Y

A continuación podremos ver como ha evolucionado la velocidad del vehículo durante la simulación:

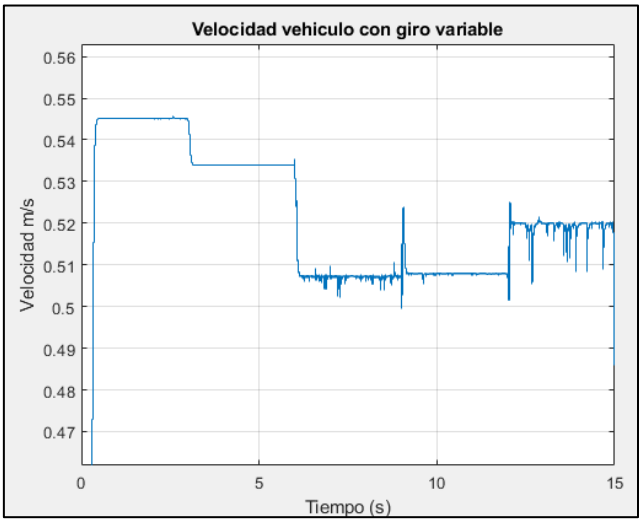


Ilustración 146:Velocidad vehículo

De la imagen anterior vemos que en los tramos en los que el giro del vehículo es más pronunciado, se produce una mayor pérdida de velocidad. Sin embargo se aprecia además que la máxima perdida de velocidad producida por el giro del vehículo se trata de alrededor de un 7% durante el giro más pronunciado. Con esto se concluye que el giro del vehículo efectivamente tiene cierta influencia sobre la velocidad del mismo, a pesar de no ser muy elevada.

A continuación, veremos como el par aplicado sobre el eje influye sobre la trayectoria recorrida por el vehículo.

Comenzaremos definiendo el par que se le aplicará al eje durante la simulación. De forma similar al caso anterior, se ha decidido aplicar un escalón variable de pares durante la simulación. En la siguiente imagen se aprecia el par aplicado durante la simulación.

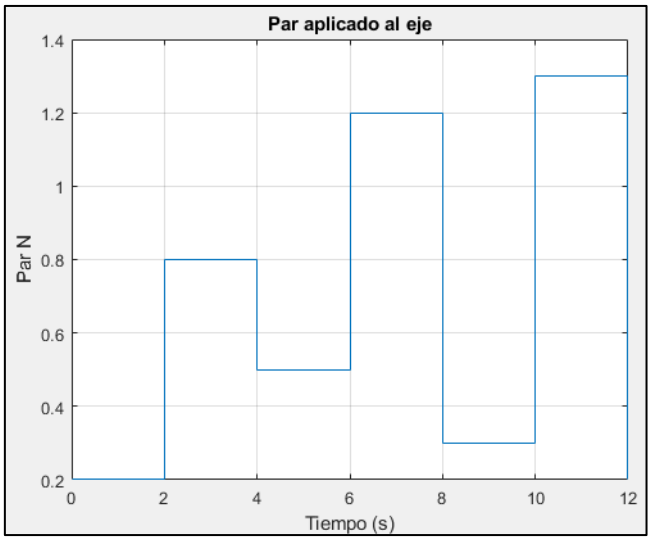


Ilustración 147: Par aplicado al eje

Sobre el servomotor, aplicaremos un giro constante de 20º. El radio medio de giro de las ruedas durante la simulación puede apreciarse en la siguiente imagen:

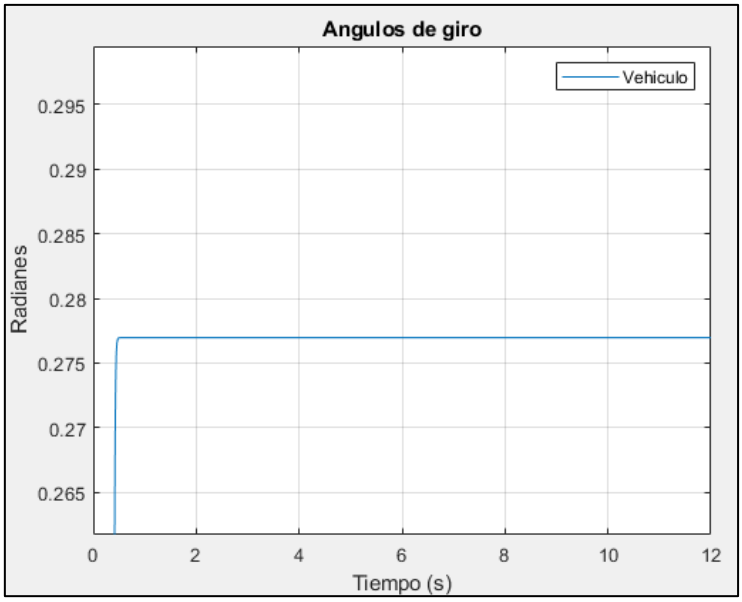


Ilustración 148: Ángulo de giro

Como puede apreciarse, el radio de giro del vehículo permanece constante durante la simulación, con un valor aproximado de 15 º.

Tras realizar la simulación podemos comprobar la trayectoria realizada por el vehículo:

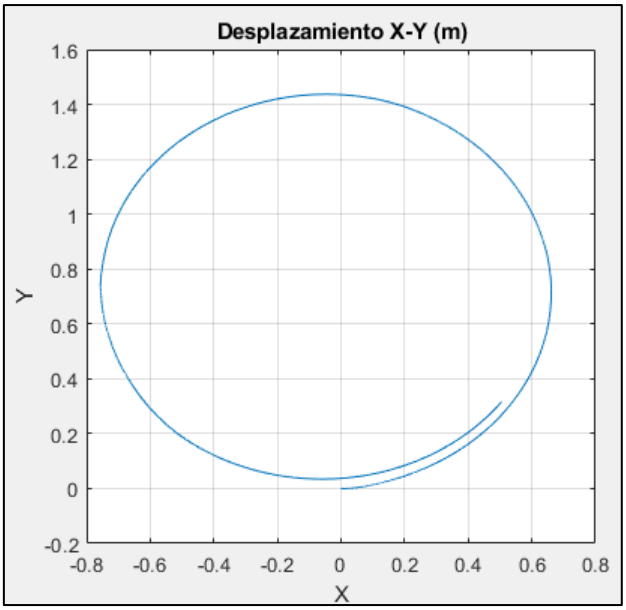


Ilustración 149: Desplazamiento X-Y

Junto a la velocidad del vehículo durante la simulación:

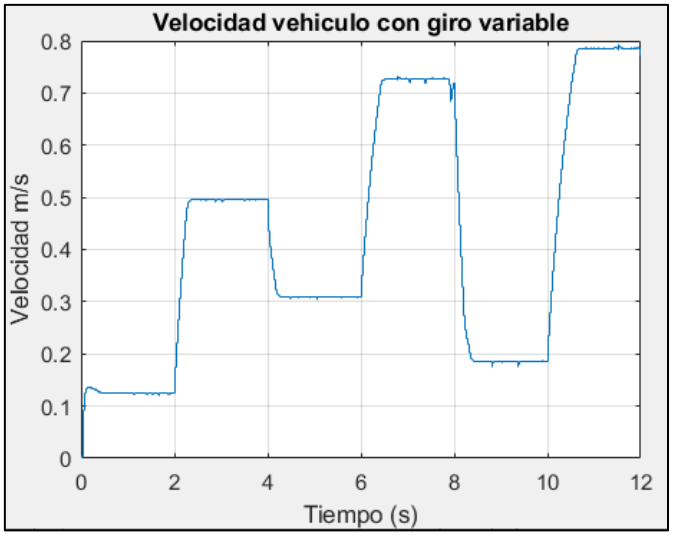


Ilustración 150: Velocidad vehículo

De la gráfica del desplazamiento del vehículo, puede apreciarse como a pesar de los cambios bruscos de velocidad, el vehículo logra mantener la trayectoria durante la simulación, más adelante se comprobará la influencia de un par en el eje variable en el seguimiento de trayectoria.

3.3.2 Control PI velocidad

A continuación se mostrará la obtención del controlador PI que nos permitirá controlar la velocidad del vehículo. Dicho controlador se diseñará para el vehículo desplazándose en dirección recta. Para comprobar el adecuado funcionamiento del mismo, comprobaremos su funcionamiento durante el giro y su comportamiento ante perturbaciones.

Un primer detalle en el que conviene centrarse es en el hecho de que si la acción de control aplicada al vehículo resulta demasiado elevada, o la pendiente de la misma resulta demasiado elevada, las ruedas del vehículo pueden deslizar sobre la superficie sobre la que se desplaza, empeorando con ello el control del vehículo. Esto se puede apreciar en la imagen inferior, en el que la velocidad real del vehículo y la velocidad teórica sin deslizar de las ruedas traseras no coinciden hasta alcanzar el medio segundo de simulación.

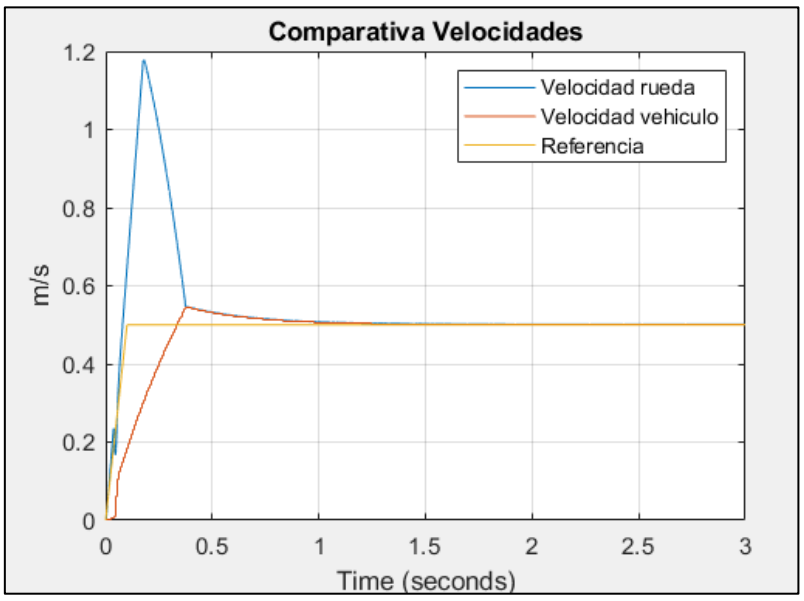


Ilustración 151: Velocidad ruedas-vehículo

Debido a ello limitaremos tanto la pendiente máxima que puede tener la señal de control así como su valor máximo durante la simulación. En base a los resultados obtenidos en el apartado anterior y a las limitaciones reales del motor eléctrico escogido, se ha decidido limitar el par máximo aplicable al vehículo a 1.5 Nm y la pendiente máxima de la señal de acción de control a m=3.

Tras realizar varias simulaciones y ajustes, se ha logrado alcanzar un controlador eficaz capaz de lograr que el vehículo alcance la velocidad deseada. Este controlador se caracteriza por un PI con las siguientes características:

$$K_P = 5 \qquad K_I = 12$$

En la siguiente imagen puede apreciarse la señal de velocidad utilizada como referencia así como la velocidad alcanzada por el vehículo:

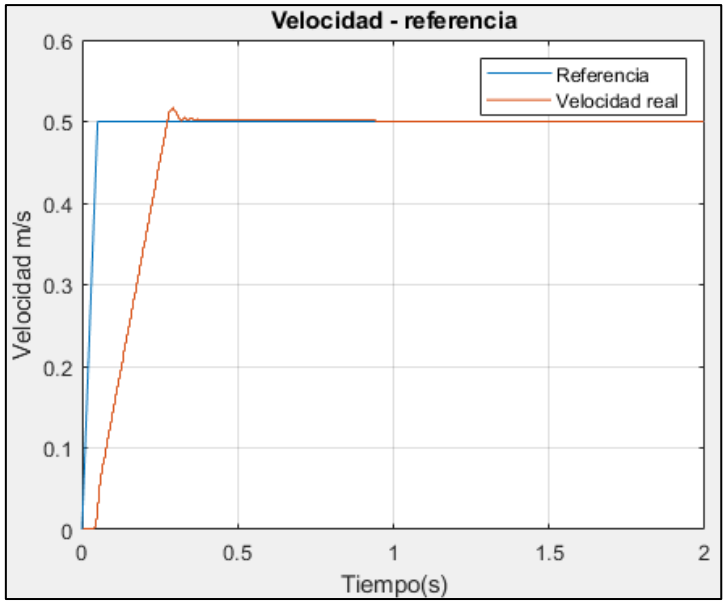


Ilustración 152: Velocidad frente a referencia

De dicha gráfica se aprecia como el vehículo logra alcanzar la velocidad deseada en poco tiempo y con un grado de sobreoscilación reducido.

A continuación veremos como se comporta dicho controlador durante el giro del vehículo y a una velocidad ligeramente superior. Para ello le aplicaremos al controlador de velocidad una referencia de 0.7 m/s. Al servomotor le aplicaremos la siguiente señal escalonada:

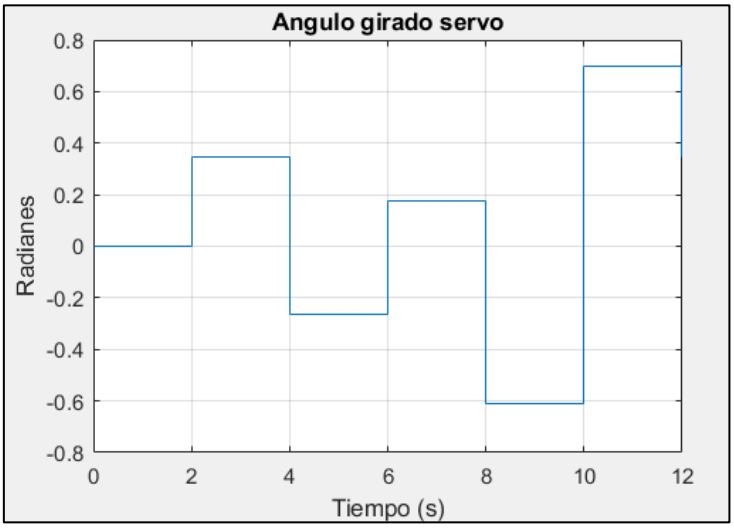


Ilustración 153 : Ángulo servomotor

Tras realizar la simulación podemos observar la velocidad del vehículo para estos ángulos de giro.

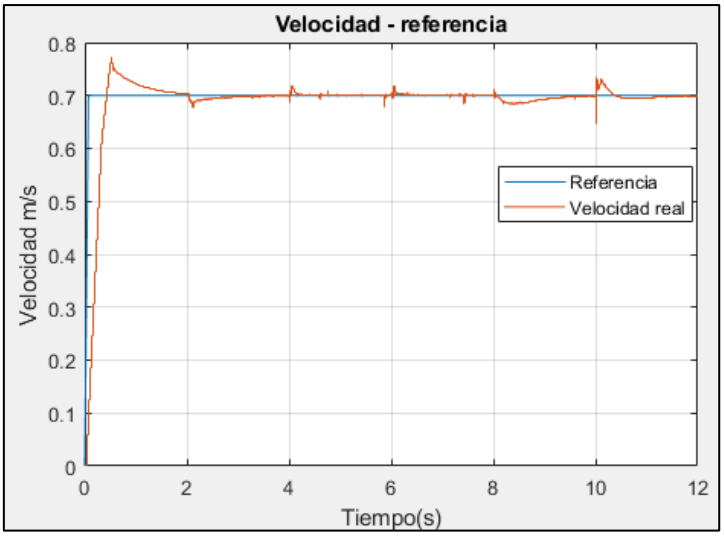


Ilustración 154: Velocidad - Referencia

De la gráfica anterior podemos deducir que el controlador logra que el vehículo siga la velocidad deseada, pero claramente se observa un cambio en la velocidad del mismo durante los cambios bruscos en el ángulo de giro del servomotor de dirección. Se observa además una mayor sobreoscilación al inicio del movimiento del vehículo.

Dichos cambios de velocidad podrían paliarse reduciendo la velocidad máxima de giro que se le permite al servomotor. Debido a que en el montaje real esta velocidad vendrá definida por el propio fabricante, no se procederá a modificar la velocidad de giro de dicho componente.

En la siguiente imagen pueden apreciarse la acción de control del vehículo durante la simulación:

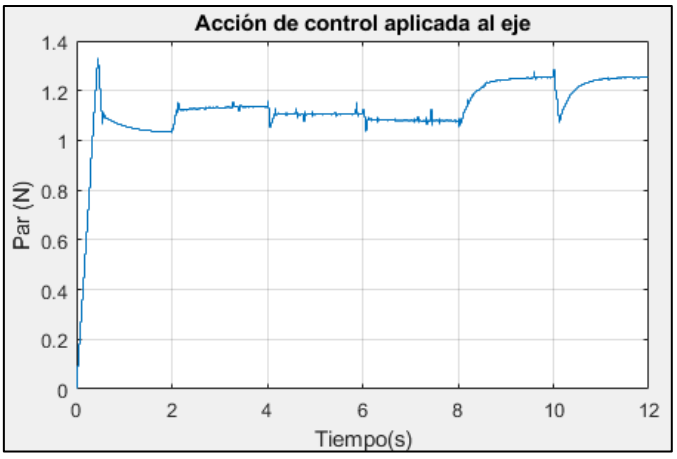


Ilustración 155: Par aplicado eje

Vemos que la acción de control aplicada sobre el eje no supera el par máximo de 1.3Nm del motor escogido salvo al inicio de la simulación en el que arranca el vehículo.

A continuación, veremos cómo se comporta el vehículo frente a perturbaciones. Para ello se aplicarán sobre las ruedas delanteras unos pares de fuerza que tratarán de frenar al vehículo

Para ello le aplicaremos a las ruedas delanteras un par que se oponga a su movimiento de -0.03Nm. este par lo aplicaremos tanto en el inicio de la simulación como en el intermedio de la misma. La perturbación aplicada sobre las ruedas tendrá con ello el siguiente aspecto:

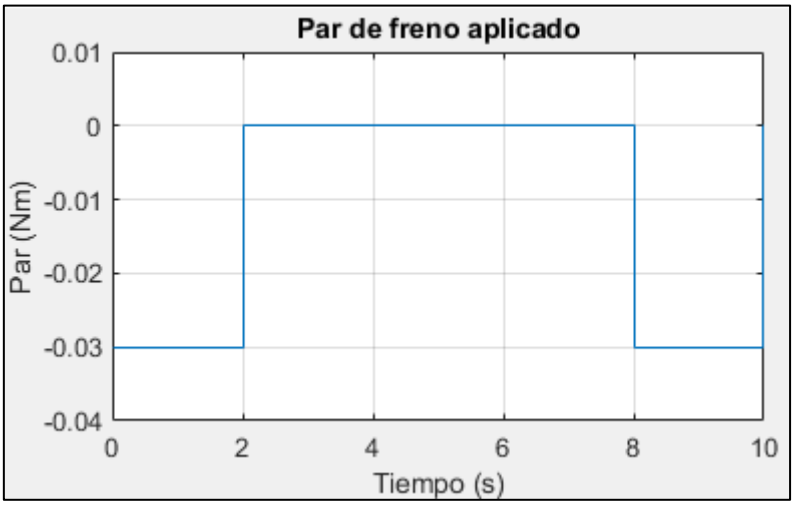


Ilustración 156: Par de freno

A continuación, veremos si el vehículo ha sido capaz de mantener la velocidad deseada durante la simulación.

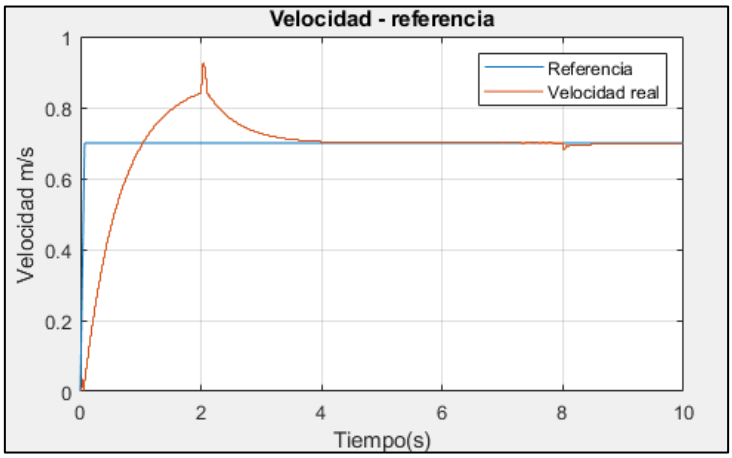
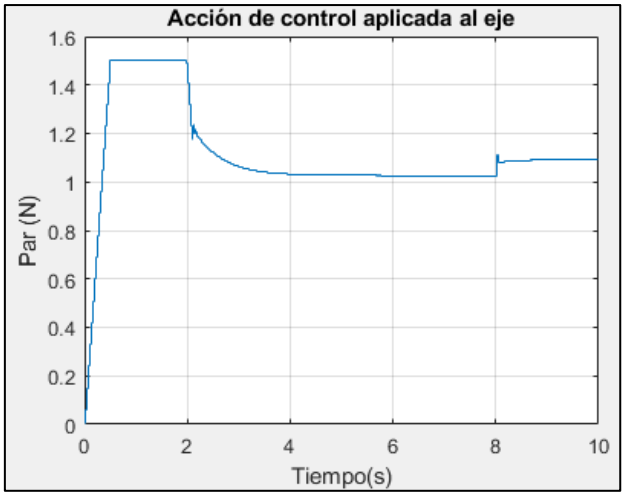


Ilustración 157: Velocidad-referencia

De la gráfica anterior puede apreciarse como el vehículo alcanza la velocidad deseada en 1 segundo, pero a pesar de ello no disminuye su velocidad hasta pasados alrededor de 3 segundos. Este problema se debe a la acumulación del error Integral que se produce durante el primer segundo de simulación. Se observa además un pico de velocidad elevado una vez deja de aplicarse el frenado sobre las ruedas delanteras.

Si observamos la acción de control del vehículo, veremos como alcanza su valor máximo al inicio de la simulación y una vez alcanzado el valor de velocidad deseado, no se reduce debido al dicho error acumulado.



Para solucionar este problema, le aplicaremos al controlador un anti-windup, que solventará el problema de la acumulación del error.

Para implementar dicho anti-windup, en el diagrama de control de velocidad cambiaremos el PI que disponíamos anteriormente por un bloque PID:

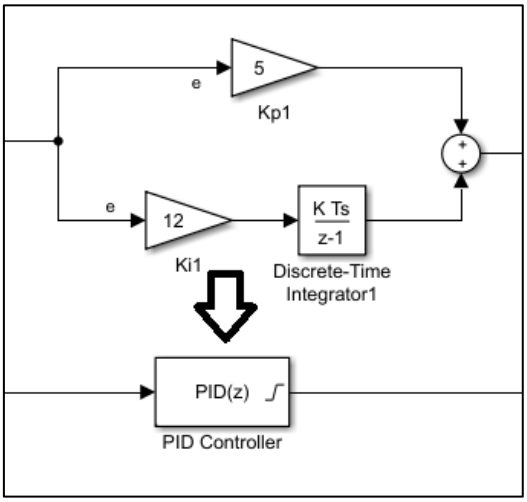


Ilustración 158: Cambio controlador velocidad

Desde dicho bloque podremos aplicar un anti-windup con la siguiente opción:

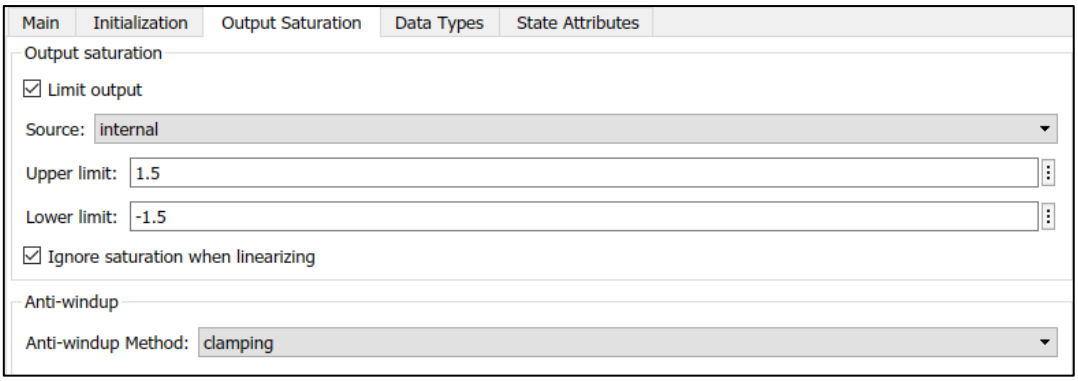


Ilustración 159: Configuración PID(z)

Tras realizar la simulación podremos observar la velocidad alcanzada por el vehículo:

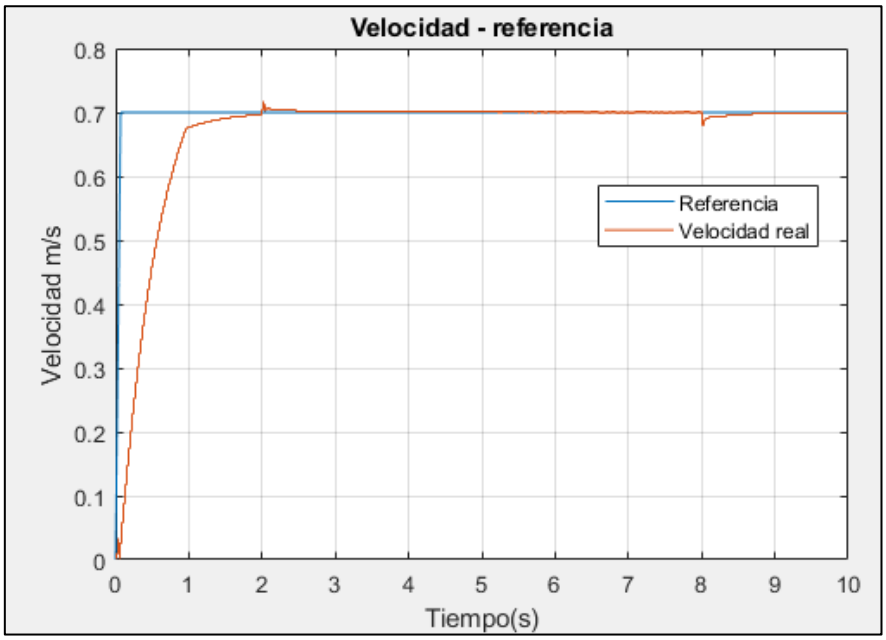


Ilustración 160: Velocidad - Referencia

Como podemos observar, el comportamiento del sistema ha mejorado considerablemente. En cuanto a la perturbación aplicada, vemos que al vehículo le lleva más tiempo alcanzar la velocidad deseada si lleva dicha perturbación. Se pueden apreciar pequeños picos de velocidad al aplicar o desaplicar el par sobre las ruedas delanteras, pero dicho pico resulta muy reducido como para considerarse como un efecto problemático.

3.3.3 Seguimiento de trayectoria

A continuación implementaremos el control de trayectoria del vehículo, el cual permitirá que el vehículo siga una trayectoria previamente definida. Para estas simulaciones mantendremos una velocidad del vehículo constante de 0.6 m/s.

Punto a punto

Comenzaremos el apartado comprobando la eficacia del sistema de control diseñado comprobando su efectividad para pasar por los puntos que definen un polígono. Este polígono ya se describió en el apartado 3.2 de trayectorias, sin embargo a modo de recordatorio se incluye una imagen del mismo:

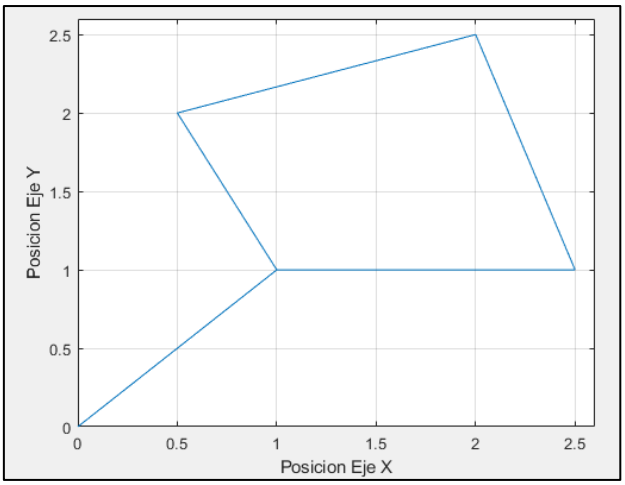


Ilustración 161: Trayectoria punto a punto

Tras realizar la simulación, podemos ver la trayectoria realizada por el vehículo:

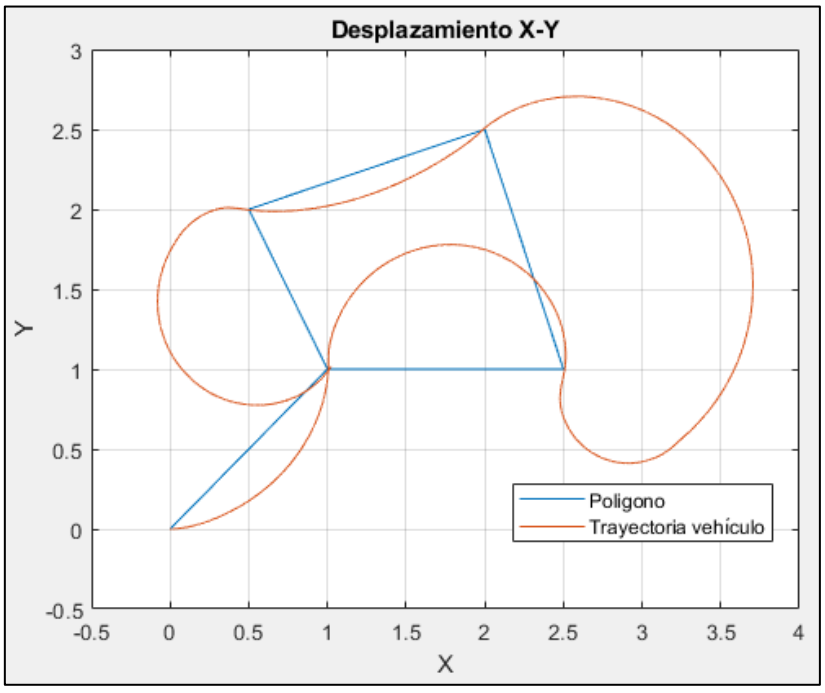


Ilustración 162: Desplazamiento X-Y

En esta podemos ver la forma en la que se desplaza el vehículo para pasar por todos los puntos que compone el polígono. Vemos que la trayectoria generada siempre trata de formar un arco circular que conecte un punto de la trayectoria con el anterior, salvo en las trayectorias de los puntos (1,2.5) a (2,2.5) y de (0.5,2) a (1,1) en los que entra en acción el compensador de ángulo de giro. Vemos que con el controlador definido por el método de Pure Pursuit el vehículo logra pasar por los 4 puntos definidos por el polígono.

Sin embargo se observa que la trayectoria realizada por el vehículo es mas larga de lo que sería necesario. Esto se solventará implementando la trayectoria mediante Curvas de Dubin.

A continuación podremos comprobar la velocidad del vehículo durante la simulación:

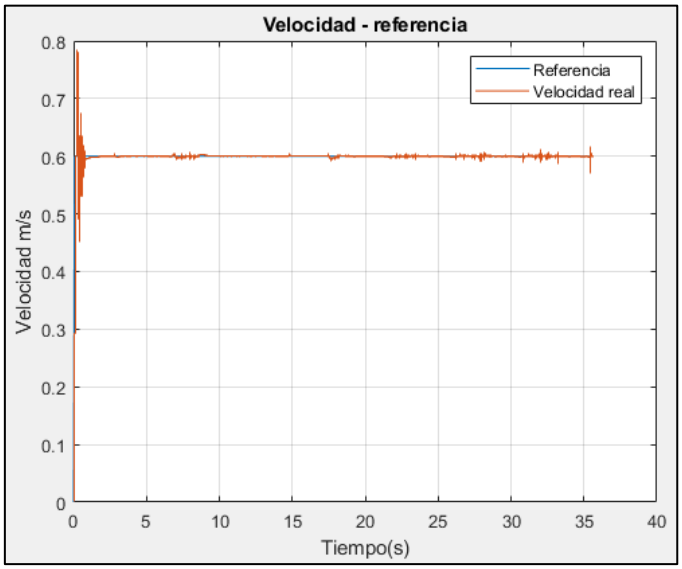


Ilustración 163: Velocidad - Referencia

Se puede observar que, salvo al inicio de la simulación, el vehículo logra mantener adecuadamente la velocidad de referencia deseada.

Curvas de Dubin

A continuación veremos la trayectoria seguida por el vehículo al definir unas curvas de Dubin que el vehículo debe ser capaz de seguir. La trayectoria a seguir se definió en el apartado 3.2.4 y se puede observar en la siguiente gráfica:

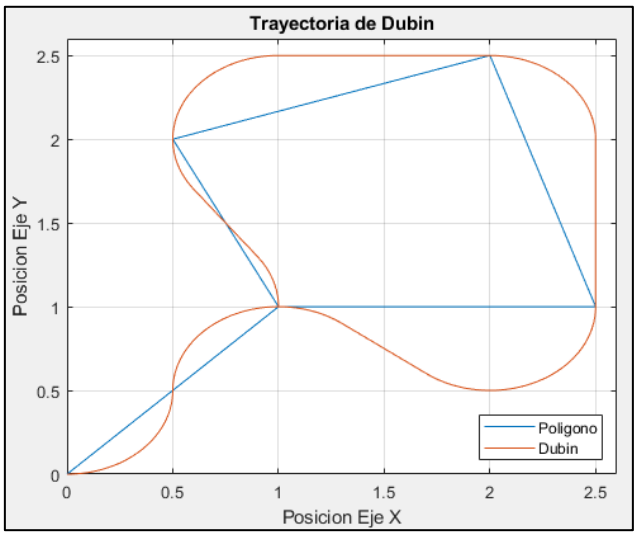


Ilustración 164: Trayectoria Dubin

Como podemos ver, esta trayectoria resulta un camino mucho mejor optimizado para pasar por los puntos que definen el polígono. A continuación, comprobaremos si el vehículo es capaz de seguir dicha trayectoria:

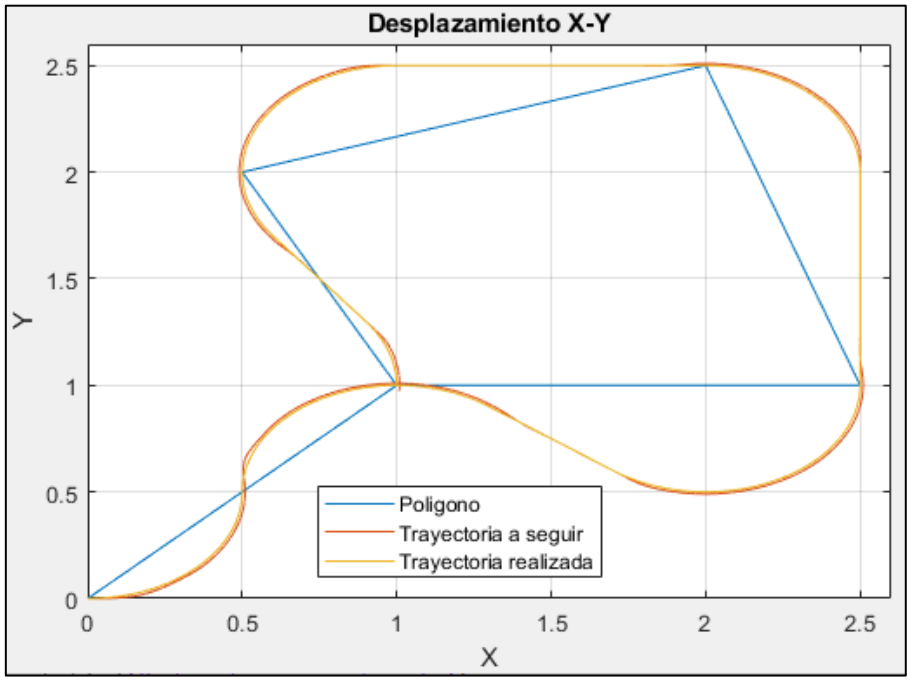


Ilustración 165: Desplazamiento realizado

Como puede apreciarse, el vehículo logra seguir la trayectoria planeada durante la simulación, teniendo una mayor dificultad en seguir la trayectoria definida durante las curvas.

En la siguiente gráfica veremos si el controlador de velocidad ha logrado que el vehículo mantenga la velocidad deseada durante la simulación.

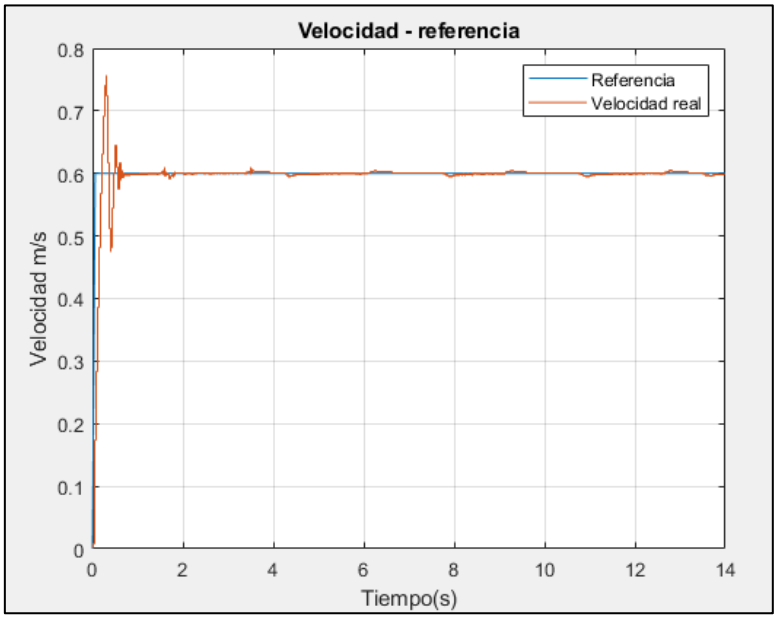


Ilustración 166: Velocidad vehículo

Puede observarse que el vehículo logra mantener adecuadamente la velocidad deseada durante la mayor parte de la simulación salvo al inicio en el que se producen unas sobreoscilaciones de velocidad.

A modo de interés general, comprobaremos la capacidad del vehículo de seguir dicha trayectoria a distintas velocidades. Primero reduciremos la velocidad de referencia del vehículo a 0.4 m/s.

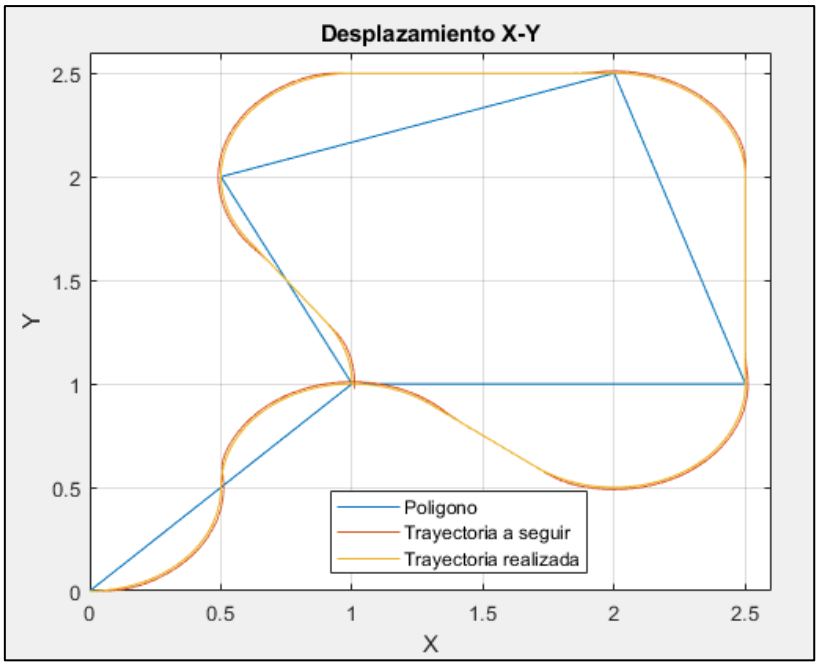


Ilustración 167: Desplazamiento X-Y

De la gráfica anterior puede comprobarse que reducir la velocidad del vehículo no mejora necesariamente la capacidad del vehículo de seguirla trayectoria definida. A continuación veremos lo que sucede si elevamos la velocidad del vehículo a 0.8 m/s:

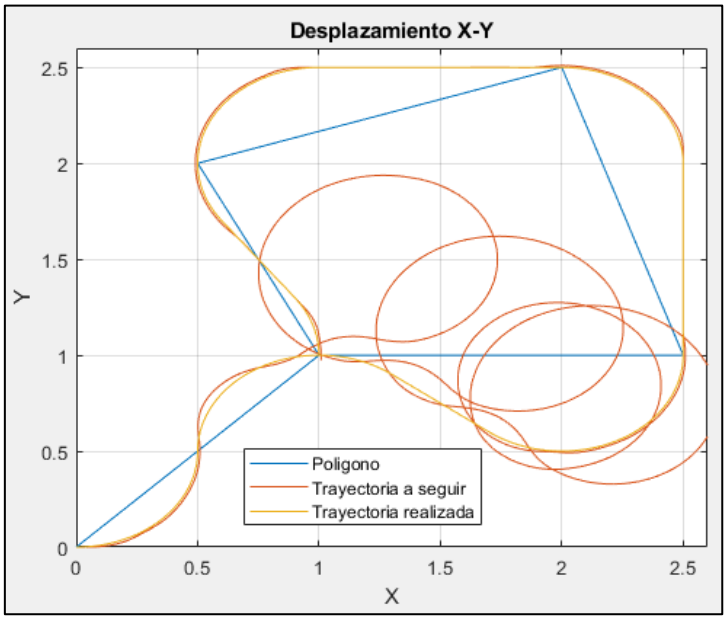


Ilustración 168: Desplazamiento X-Y

4 Implementación real

En este capítulo trataremos de describir el prototipo realizado del vehículo diseñado. Para ello se ha diseñado un modelo de vehículo con toda la parte tanto mecánica como electrónica del vehículo.

Comenzaremos el capítulo describiendo todas las características de montaje del vehículo, así como describiendo los distintos componentes y herramientas utilizadas en su montaje. Seguidamente se describirá como se ha realizado el control del vehículo mediante bluetooth, así como una breve descripción de los programas diseñados en Arduino para realizar el control del vehículo.

Cabe mencionar que este capítulo tiene una gran dependencia con el Anexo I: Planos, ya que es en donde se incluyen las características dimensionales de los componentes utilizados en el montaje.

4.1 Montaje realizado

En este primer capítulo veremos el propio prototipo diseñado así como los distintos componentes utilizados, tanto a nivel mecánico como electrónico. Se tratará de describir el proceso de montaje del vehículo, así como el conexionado de los distintos componentes electrónicos que componen el vehículo.

4.1.1 Prototipo real

En este apartado podremos ver unas imágenes del prototipo diseñado a partir del modelo realizado en el capítulo de simulación.

Comenzaremos viendo el modelo así como se diseñó en Solidworks:

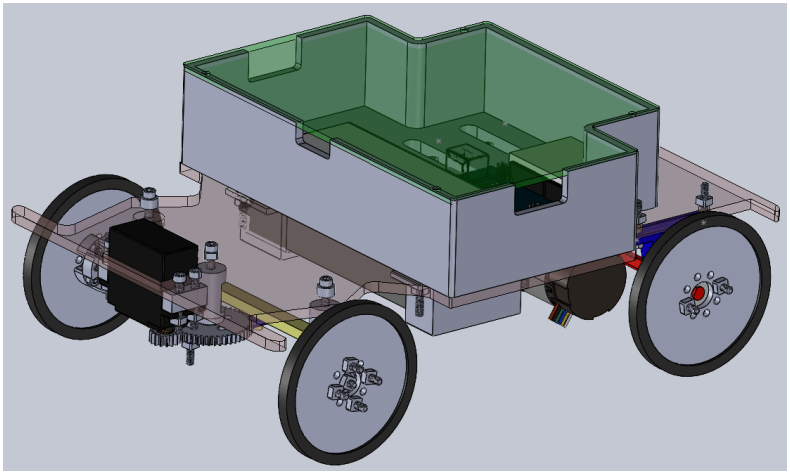


Ilustración 169:Prototipo real CAD – vista 1

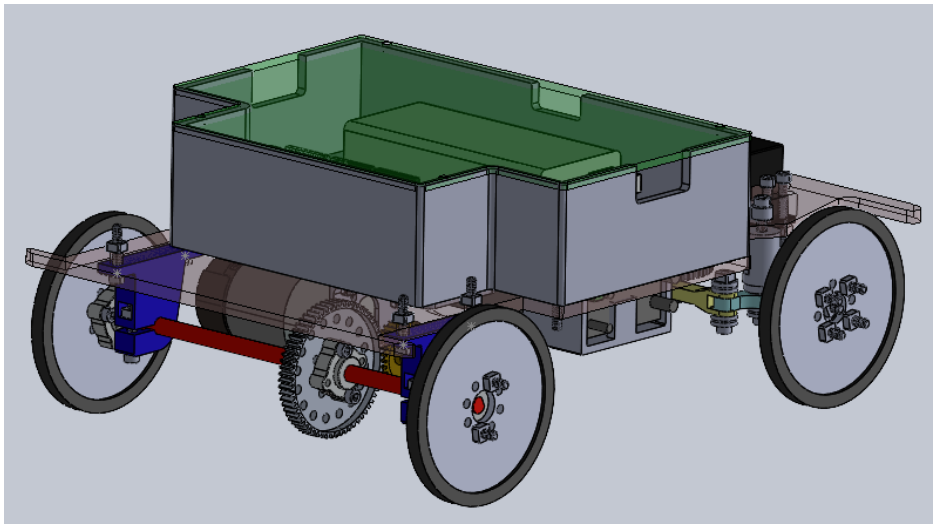


Ilustración 170: Prototipo real CAD - vista 2

Tras diseñar un modelo que fuese físicamente realizable se procedió a la compra de los distintos materiales que fueron necesarios para montar el prototipo real.

En las siguientes imágenes puede observarse el montaje final realizado:

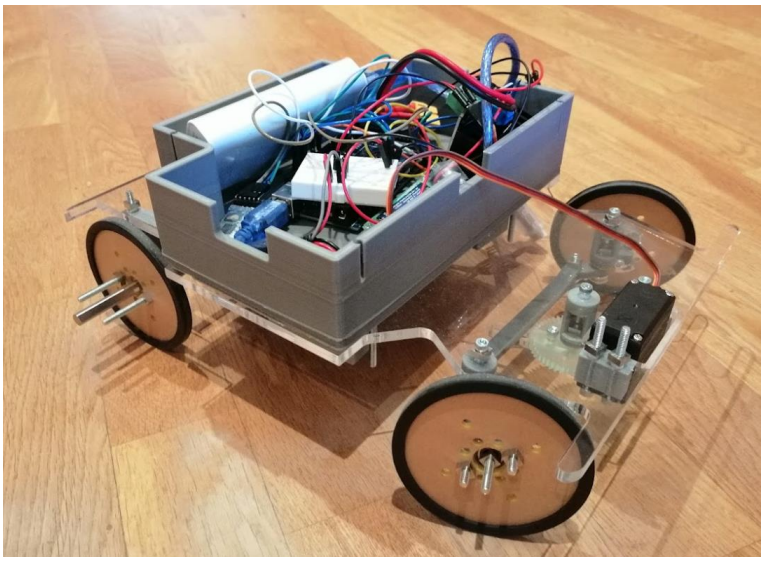


Ilustración 171: Prototipo real - Vista 1

Como podemos ver, la elección de metacrilato para la base y el diseño realizado para la carcasa de la electrónica nos permite ver en todo momento el mecanismo de Ackerman diseñado para el vehículo.

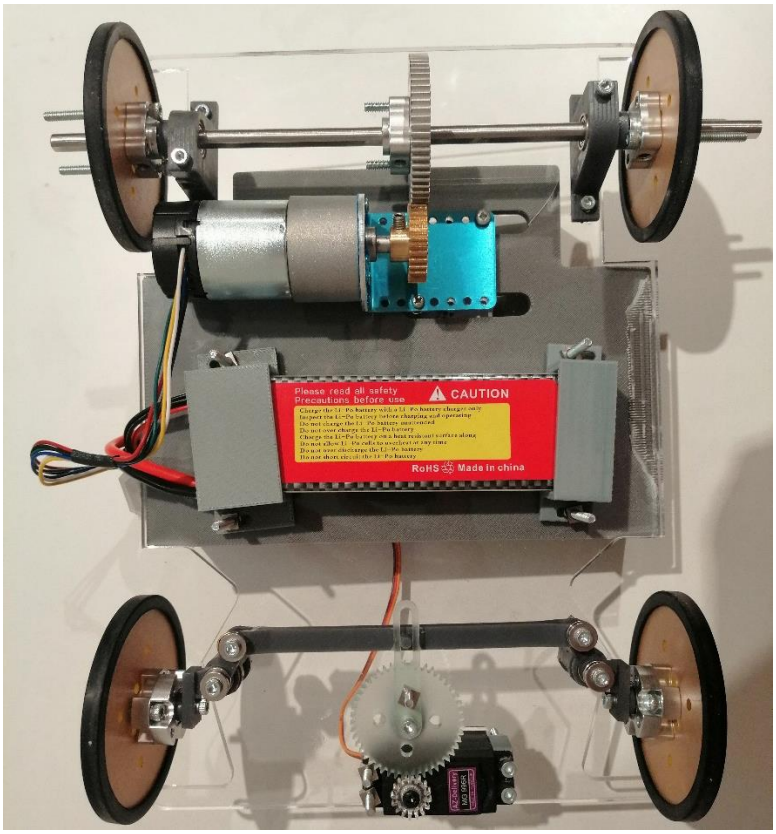


Ilustración 172: Prototipo real - Vista 2

En las siguientes imágenes puede apreciarse con mayor detalle cómo se ha realizado el montaje de la parte del mecanismo de Ackermann:

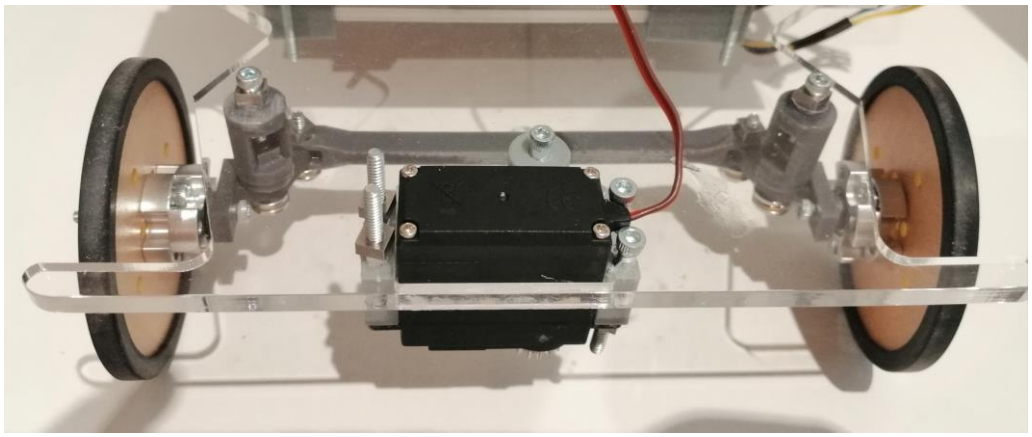


Ilustración 173: Prototipo real - Vista dirección 1

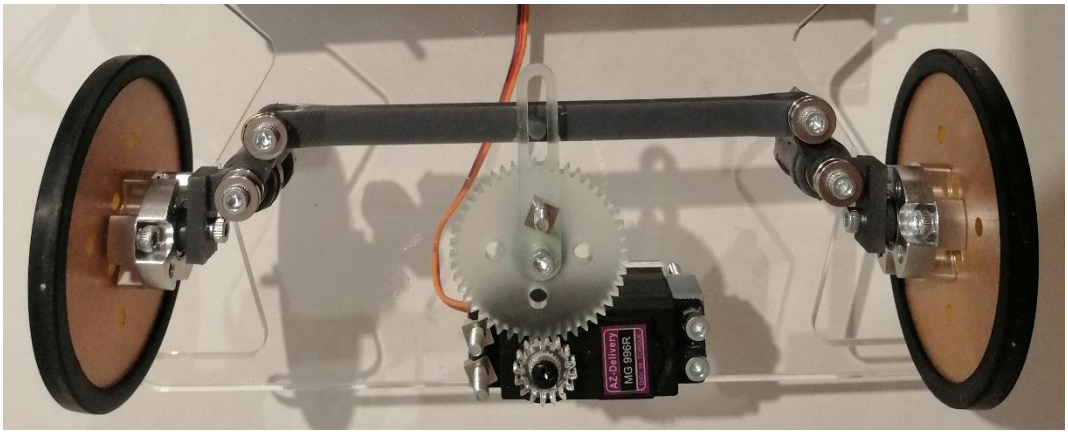


Ilustración 174: Prototipo real - Vista dirección 2

Cabe mencionar que el montaje del vehículo no vino sin dificultades añadidas, debido en gran parte al elevado tiempo de envío de algunos de los componentes. En el apartado 4.5 se detallan los problemas encontrados durante el montaje y la solución adoptada para solventar dichos problemas.

4.1.2 Lista de componentes

El objetivo de este apartado es describir todos los componentes que forman parte del vehículo, ya sean ruedas, cables o piezas impresas, así como realizar una breve descripción de su función realizada.

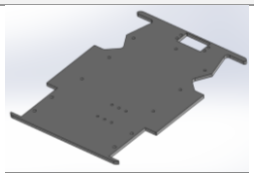
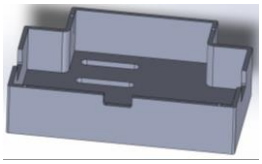
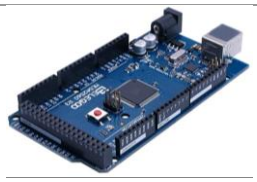
Cabe mencionar, que debido a la complejidad constructiva del vehículo y los componentes disponibles, se decidió realizar el vehículo con componentes de la marca Actobotics, debido a su elevada variedad de piezas disponibles. Un problema que esta decisión supuso, fue la utilización de un estándar que no fuese el estándar ISO. En el apartado de montaje describiremos los problemas que surgieron a raíz de esta decisión.

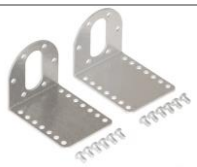
No se entrará en detalles técnicos de los componentes ya que se considera que esto queda cubierto en los Anexos I y III (Planos y Datasheet).

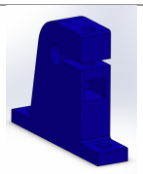
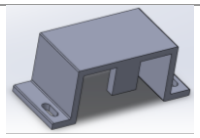
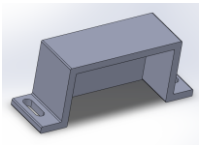
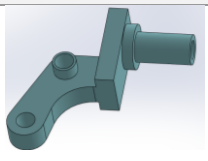
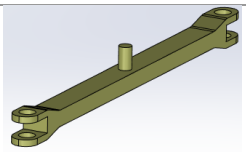
Para facilidad de comprensión, dividiremos esta tabla en 3 secciones, siendo 2 de ellas las secciones de tracción y dirección que se habían venido describiendo anteriormente, y una tercera a modo general para describir cualquier componente que pueda formar parte de ambos grupos o ninguno.

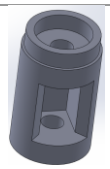
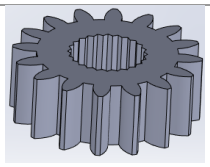
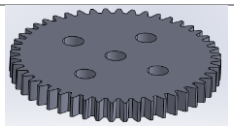
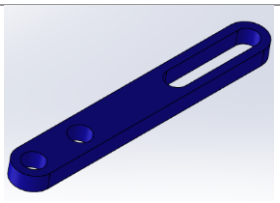
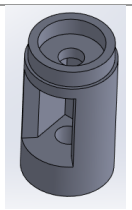
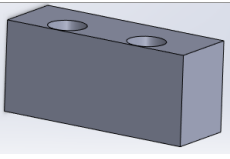
Los componentes descritos a continuación podrán ser o bien piezas comerciales, creadas mediante impresión 3D u obtenidas mediante corte laser de metacrilato. En la tabla se incluirá una letra entre paréntesis indicando a que grupo pertenece cada componente:

- M (metacrilato)
- C (comercial)
- I (Impresa)

Componente	Función	Imagen
General		
Base del vehículo (M)	Componente creado a partir de metacrilato cuya función es la de realizar de 'carrocería' del vehículo, sujetando todos los componentes del vehículo.	
Ruedas de precisión de 3" - Actobotics (C)	Estas ruedas son las que realizan el contacto del vehículo con el suelo y su función es la de asegurar el adecuado movimiento del vehículo.	
Tornillos 6-32 de longitud ¾" y 1.5" (C)	Estos tornillos se encargarán, junto a las tuercas, de sujetar las distintas piezas que montaremos sobre la base del vehículo o entre ellas	
Tuercas cuadradas 6-32 (C)	Este tipo de tuercas, caracterizadas por tener una forma cuadrada, se utilizarán para sujetar las piezas que requieran montarse mediante tornillos	
Carcasa electrónica (I)	Esta carcasa se ha diseñado con el propósito de albergar toda la electrónica que requiera el vehículo tal y como drivers, microcontrolador, o módulo bluetooth.	
PowerBank 5V (C)	Batería externa recargable cuya función original es la de cargar teléfonos móviles. En nuestro vehículo se encargará de proporcionarle tensión a toda la parte electrónica del vehículo salvo a los motores	
Microcontrolador Arduino Mega 2560 (C)	Controlar el vehículo actuando de intermediario entre todos los componentes electrónicos del vehículo desde el motor hasta el módulo bluetooth.	
Driver IBT-2 (C)	Permite el control completo del motor de 12 V mediante un microcontrolador que trabaja a 5 V.	

Módulo Bluetooth HC-05 (C)	Permitir la comunicación inalámbrica del vehículo ya sea mediante un mando de videoconsola o un teléfono móvil	
Cables varios (C)	Diversos tipos de cables utilizados para realizar la conexión entre los distintos componentes electrónicos del vehículo.	
Regulador de tensión (C)	Componente utilizado para reducir la tensión de 12 V de la batería grande a 6 V para poder conectar el Servomotor.	
Tracción		
Motor 12 V con encoder - Pololu (C)	Motor principal del vehículo que asegurara el desplazamiento del mismo. Dispone de un encoder para medir tanto la velocidad de desplazamiento del vehículo como la distancia desplazada, de forma aproximada.	
Soporte Motor Pololu (C)	Componente encargado de sujetar el motor de 12 V a la base del vehículo.	
Eje tipo D de 12” - ¼” – Actobotics (C)	Eje principal del sistema de tracción. Esta barra se encarga de transmitir la potencia de los engranes a las ruedas del motor. La forma en ‘D’ de la barra facilita la transmisión de potencia de un extremo de la barra a otro.	
Piñon de 32D Ø6mm - Actobotics (C)	Engranaje que se monta en el motor de 12V que transmite el giro del motor al engranaje montado sobre el eje principal del sistema de tracción.	
Engranaje Aluminio 64T 0.5” – Actobotics (C)	Engranaje que se monta sobre el eje principal del vehículo que transmite el giro del motor al propio eje sobre el que se encuentra montado.	

Set de montaje de tornillo 0.77” Ø ¼” (C)	Componente cuya función es la de sujetar tanto las ruedas del vehículo como el engranaje de 64 dientes al eje principal del vehículo. Dispone de un tornillo prisionero que nos permite montar y fijar el set sobre el eje.	
Soporte Eje (I)	Este componente se encarga de mantener sujeto a la base del vehículo el eje principal de tracción del vehículo. Dispone de un agujero con espacio para alojar un rodamiento del bolas sobre el que se apoyará el propio eje. Para el montaje dispone de 2 agujeros en su parte inferior para el montaje de los tornillos.	
Rodamiento radial de bolas ¼” x ½” – Actobotics (C)	Rodamiento montado sobre el soporte del eje principal del vehículo y el propio eje, permitiendo a este último girar con un nivel de rozamiento lo más reducido posible	
Batería Lipo 3500mAh 25A 11.1V (C)	Esta batería será la que utilizaremos para alimentar a los 2 motores del vehículo.	
Cargador batería Lipo B3 Pro (C)	Este cargador tiene la función de cargar la batería mencionada anteriormente	
Soportes batería 12V (I)	Estas 2 piezas se montan en la parte inferior del vehículo y tienen la función de sujetar la batería de 12 V a la base del vehículo	 
Dirección		
Bieleta dirección (I)	Permitir la conexión de la rueda con el resto del sistema de dirección, así como de transmitir el giro del servomotor de direccción a las ruedas directrices. Le corresponde una a cada rueda.	
Barra de dirección (I)	Actua como unión entre ambas bieletas y transmite el movimiento de giro a las mismas mediante el saliente central que dispone	

Servomotor MG996R (C)	Este es el dispositivo sobre el que actuaremos mediante el microcontrolador para poder dirigir el vehículo	
Buje de rodamiento de bolas (C)	Componente que utilizaremos para montar las ruedas directirces sobre las bieletas del sistema de dirección. El rodamiento permitirá a la rueda girar libremente.	
Rodamientos axiales F5-10M (C)	Permiten el giro libre de las bieletas sobre los apoyos que se montarán sobre la base.	
Soporte bieletas (I)	Se monta directamente sobre la base del vehículo y sirve a modo de apoyo de las bieletas del sistema de dirección	
Piñon 16 D (M)	Este engrane se monta sobre el servomotor y forma parte de la cadena mecánica que transmite el giro del servomotor a las ruedas directrices	
Engrane 48D (M)	Engrane que desplaza el brazo de dirección e una dirección u otra según el ángulo de giro del piñon montado sobre el servomotor	
Brazo de dirección (M)	Este componente se monta fijo sobre el engrane mencionado anteriormente. En la guía que dispone el componente, se introduce el saliente de la barra de dirección, permitiendo con ello el desplazamiento de la barra de dirección al girar el servomotor.	
Soporte Engrane 48D (I)	Este componente tiene la función de permitir el apoyo del engrane de 48D junto al brazo de dirección sobre la base. Incluyendo un espacio para albergar un rodamiento axial, facilitando de esta forma el giro del sistema de dirección.	
Soporte Servomotor (I)	Estos bloques sirven para poder montar el servomotor a una altura que permita que el mismo engrane adecuadamente con el engranaje del brazo de dirección	

4.2 Control manual del vehículo

En este apartado se pretende describir como se ha implementado el control manual del vehículo.

Este método de control trata de controlar el avance y giro del vehículo directamente mediante una señal, la cual será enviada al vehículo por medio de una conexión Bluetooth. El objetivo es que el usuario pueda conectarse al vehículo a través de cualquier dispositivo con Bluetooth y controlar independientemente el giro de las ruedas directrices, así como el avance o retroceso del vehículo.

En este documento se presentan 2 métodos de control manual: mediante un smartphone o un mando de videoconsola.

Esta tarea va a requerir por un lado el hardware para que el vehículo pueda recibir las señales enviadas y por otro lado el software que se encargará de tratar las señales recibidas.

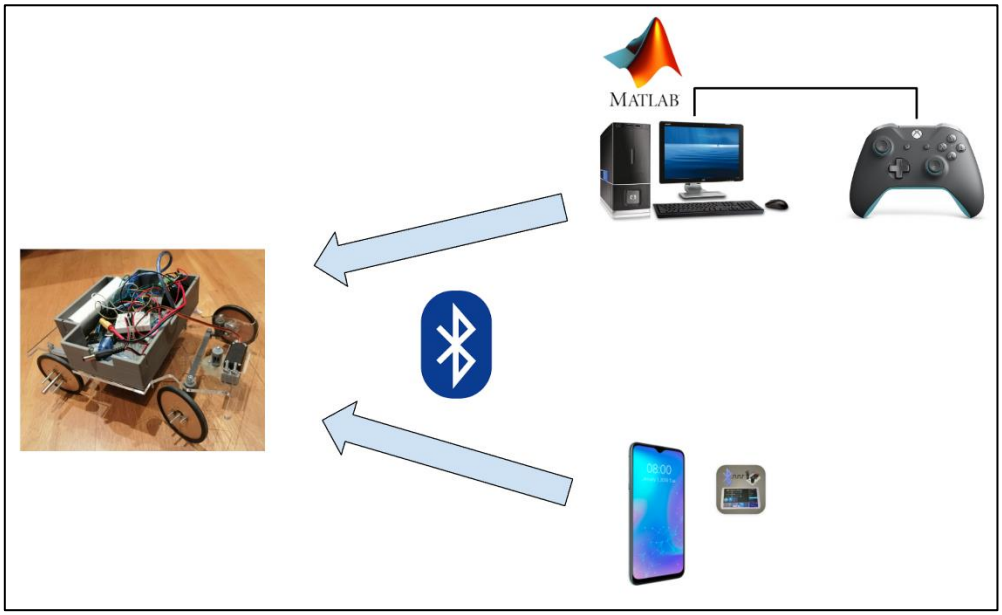


Ilustración 175: Diagrama funcionamiento

4.2.1 Hardware

A nivel general requeriremos de una placa de Arduino para procesar las señales recibidas y actuar sobre los motores y un módulo Bluetooth HC-05 para recibir las señales de los controladores y enviarlas al microcontrolador.

En el caso de que queramos controlar el vehículo mediante un smartphone, solo necesitaremos un smartphone con conexión Bluetooth y una aplicación que pueda conectarse al módulo Bluetooth utilizado.

Para el caso de control mediante mando, se ha utilizado un mando de Xbox, un cable para conectar el mando al ordenador y un ordenador que dispone de conexión bluetooth.

4.2.2 Software

A nivel de software tendremos el Arduino IDE que usaremos para programar el Arduino que recibirá los datos del módulo bluetooth, y en función de los datos recibidos realizará una acción u otra sobre los motores del vehículo. En el Anexo II, se puede observar el código diseñado para realizar el control del vehículo.

Para el control por smartphone nos aprovecharemos de la aplicación 'Bluetooth Electronics' que nos permite conectarnos al módulo HC-05 y enviarle unas señales que podremos programar mediante la propia aplicación. En la imagen inferior se puede apreciar una captura del control creado para el vehículo, compuesto por 2 barras deslizantes:

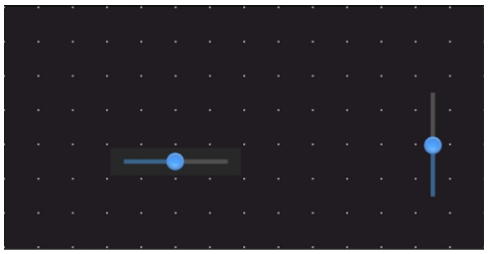


Ilustración 176: Control Smartphone

A la hora de Implementar el controlador PI en el vehículo, se decidió ampliar la funcionalidad del programa de forma que permita visualizar la velocidad actual del vehículo así como su referencia. Esto se documenta en el apartado 4.4 de la memoria.

Por otro lado, tenemos la parte de software que lee la entrada del mando Xbox y envía dichas lecturas por bluetooth al vehículo. Para esta parte podríamos usar una infinidad de programas, pero se ha decidido utilizar un script de Matlab para simplificar el proceso y no requerir instalar programas adicionales en el ordenador.

4.2.3 Principio de funcionamiento

El principio de funcionamiento de este método de control se basa en el envío de mensajes cortos que le indican a la placa de Arduino las acciones que debe realizar.

Estos mensajes tienen 2 formatos:

- "S" + N + "A": Control del Servomotor en el que N es un número entero entre 35 y 160, que indica los grados que queremos que giren las ruedas directrices del vehículo. Siendo 90 el punto central de giro de las ruedas.
- "M" + N + "A": Control del motor principal en el que la N es un número entero entre 0 y 255 que indica en qué dirección queremos que gire el motor y a que velocidad. La velocidad nula se alcanza al enviar un valor de 127.

En el control por mando Xbox se utilizan el joystick izquierdo para controlar el giro del vehículo y los gatillos traseros para controlar la velocidad y avance/retroceso del vehículo.

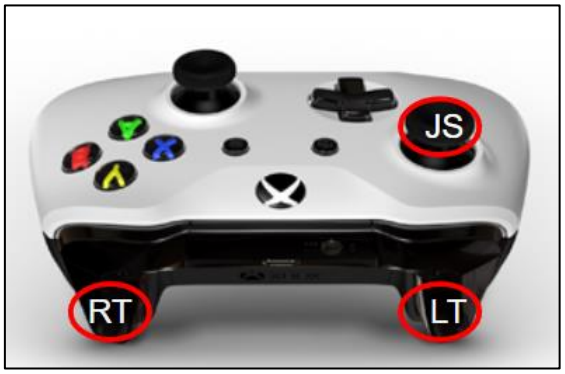


Ilustración 177: Control mando Xbox

En el control diseñado, el botón 'RT' controla el avance del vehículo mientras que LT controla el retroceso del mismo.

En el control mediante la aplicación bluetooth para smartphone que se describió anteriormente el slider vertical controla la velocidad del vehículo mientras que el slider horizontal se encarga de controlar la dirección de giro de las ruedas.

4.3 Montaje eléctrico

En el siguiente capitulo se muestra todo lo relacionado al montaje electrónico-eléctrico del vehículo. Primeramente se justificará la elección de 2 baterías para proporcionarle potencia al sistema. Seguidamente se describirá brevemente el esquema de montaje realizado así como una descripción del conexionado de los componentes de mayor importancia.

4.3.1 Cálculos potencia

Para el montaje deseado se requiere preferiblemente de una batería de 12 voltios, ya que de esta forma aprovecharemos al máximo el motor principal. De la hoja de características del motor vemos que la corriente máxima que el motor puede requerir a 12 V es de 5.5 A. Debido a esto se ha seleccionado una batería Lipo de 11.1V y una descarga de 25 A.

El servomotor escogido tiene una intensidad de funcionamiento entre 500 y 900 mA. Esta intensidad resulta demasiado elevada como para poder conectarla directamente a la placa de Arduino, debido a ello utilizaremos un regulador de tensión para reducir la tensión de la batería a una tensión aceptable para el servomotor (6V).

En teoría con la misma batería deberíamos ser capaces de conectar la placa de Arduino junto al módulo HC-05. Sin embargo, tras realizar varias pruebas en las que el módulo HC-05 se desconectaba, perdiendo con ello la conexión al sistema de control manual, se decidió utilizar una batería aparte para la placa de Arduino y el módulo Bluetooth, como podemos ver en el diagrama del siguiente apartado.

4.3.2 Esquema montaje

En la siguiente imagen puede apreciarse el montaje eléctrico realizado para el vehículo:

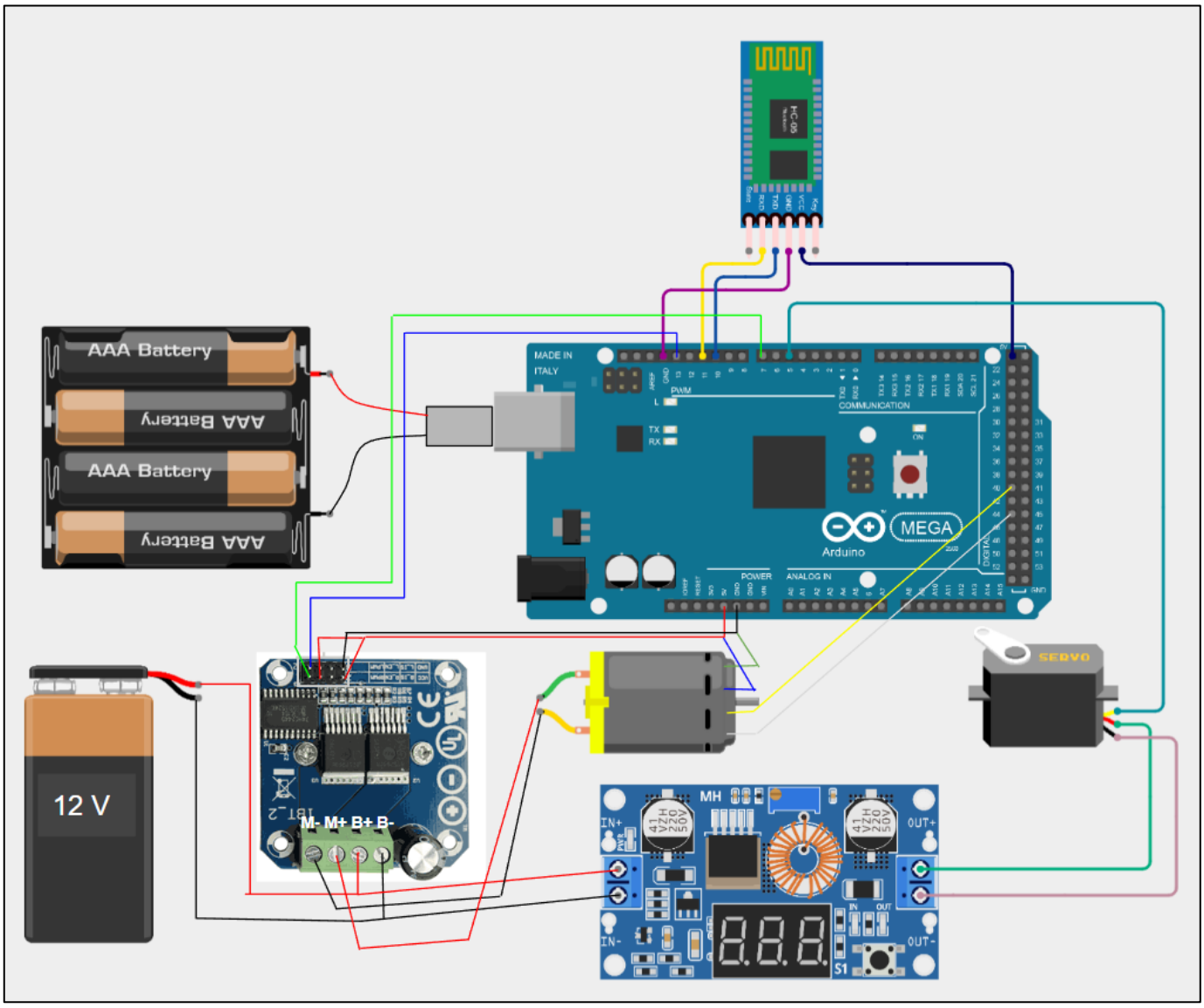


Ilustración 178: Esquema eléctrico vehículo

A continuación realizaremos una breve descripción de como se han conectado los componentes de mayor relevancia a la placa de Arduino así como la justificación de dichas conexiones.

<u>Módulo Bluetooth HC-05</u>

Este componente realiza una conexión entre un dispositivo externo y la placa de Arduino. El módulo dispone de las siguientes conexiones:

- GND: Conectado a tierra
- VCC: Conectado a 5V
- RX: Conectado al pin 11
- TX: Conectado al pin 10

El pin TX tiene la función de transmitir los datos que llegan del ordenador o smartphone a la placa Arduino para su posterior procesamiento.

El pin RX del módulo es el encargado de recibir los mensajes que la placa de Arduino envía al módulo que se transmitirá por bluetooth al dispositivo conectado.

En la placa de Arduino definiremos el pin 10 como pin de Recepción(RX) y el pin 11 como pin de transmisión (TX).

Driver-IBT2

Este Driver se compone por un lado de la electrónica relacionada con el control del motor y por otro lado de la electrónica de potencia que se conecta a la fuente de alimentación y al motor. En la siguiente imagen podemos ver los distintos pines que contiene dicho driver.

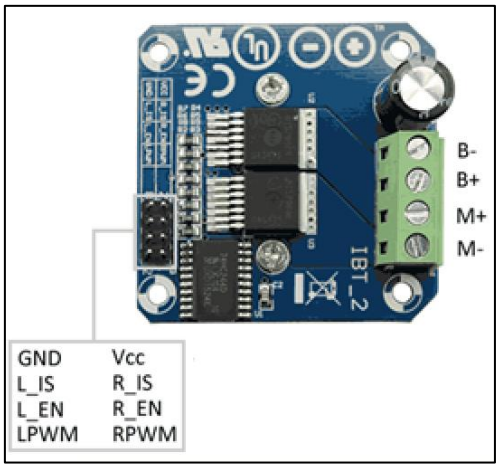


Ilustración 179: Conexión driver IBT-2

El driver permite controlar la dirección de giro del motor mediante los pines de R_EN y L_EN y la velocidad de giro del mismo mediante las entradas de señal PWM, LPWM y RPWM.

La parte derecha del driver se componen de los conectores B y M, los cuales se conectan respectivamente directamente a la fuente de alimentación y al motor que se desea alimentar.

En la parte izquierda del driver se encuentran los pines relacionados con el control del driver. Estos pines se conectan de la siguiente forma:

- GND: conexión a tierra común con la placa de Arduino
- Vcc: Conectado a 5 V
- L_IS y R_IS: Sin conectar
- L_EN y R_EN: Estos pines activan o desactivan el giro del motor conectado en una dirección u otra. Para facilitar el funcionamiento del programa, conectaremos estos pines a una tensión de 5V, quedando con ello activos constantemente.
- LPWM y RPWM: Estas entradas son las que controlan la velocidad del motor. Debido al modo de funcionamiento escogido, solo podremos enviar una señal PWM a uno de estos pines a la vez. De esto se encargará el propio código diseñado para controlar el vehículo.

Motor 12 V

Este motor dispone de un total de 6 conectores distintos, 2 para aportar tensión al propio motor y 4 para el encoder.

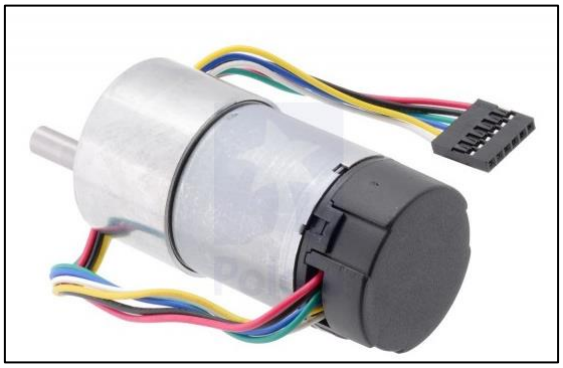


Ilustración 180: Motor

Los distintos conectores se distinguen por el color del cable al que se conecta de la siguiente manera:

- Rojo: Este conector se conecta al borne positivo de la batería de 12 V y aporta la tensión requerida para el funcionamiento del motor
- Negro: conexión a tierra del motor.
- Verde: Conexión a tierra del encoder
- Azul: Conexión de 5 V para el correcto funcionamiento del Encoder
- Amarillo: Lectura A del encoder, conectado al pin 2 del Arduino
- Blanco: Lectura B del encoder, conectado al pin 3 del Arduino

Para el adecuado funcionamiento de la lectura del encoder, los pines de lectura se definirán como interrupciones a la hora de programar el microcontrolador Arduino.

4.4 Implementación PI

A continuación, describiremos como se ha implementado el controlador PI de velocidad en el vehículo diseñado.

Debido a la estabilidad de conexión, este apartado se realiza en su totalidad mediante un smartphone y la aplicación Bluetooth Electronics.

Para medir la velocidad del vehículo aprovecharemos el encoder que dispone el motor del vehículo. Este encoder es capaz de medir 4480 pulsos por revolución del motor, teniendo en consideración este dato, podremos obtener la relación entre los pulsos medidos por el encoder y la velocidad del vehículo.

Las ruedas del vehículo se encuentran montadas sobre un eje que dispone de un engrane de 64 dientes que engrana con el motor, que dispone de un piñón de 32 dientes. Esto quiere decir que por cada vuelta que giran las ruedas traseras, el encoder leerá un total de 8960 pulsos.

Las ruedas escogidas tienen un diámetro de 3" (7.62 cm) con lo que por cada vuelta que complete la rueda el vehículo se habrá desplazado:

$$7.62\ cm \cdot \pi \cong 23.94\ cm$$

De este valor podemos obtener la relación directa entre los pulsos medidos por el encoder y el desplazamiento lineal del vehículo:

$$\frac{8960\frac{pulsos}{vuelta}}{23.94\frac{cm}{vuelta}} = 374\frac{pulsos}{cm} = 37400\frac{pulsos}{m}$$

El propio Arduino nos permite obtener el tiempo transcurrido entre un período y el anterior. De esta forma si tras un período de funcionamiento de 10 ms, la placa de Arduino ha detectado un total de 200 pulsos, podremos obtener la velocidad del vehículo durante ese período:

$$\frac{200\ pulsos}{374\ pulsos/cm} \cdot \frac{1}{0.01s} = 53{,}47\frac{cm}{s} \approx 0.5\ m/s$$

La programación de dicho controlador se puede comprobar en el Anexo II: Códigos.

La velocidad requerida en cada momento se enviará mediante el control manual del vehículo.

Cabe mencionar que la velocidad real del vehículo no habrá forma de medirla ya que la lectura de velocidad obtenida en base al encoder no toma en consideración el deslizamiento que pueda producirse entre las ruedas del vehículo y el suelo o el desfase de velocidad producido al girar el vehículo.

Inicialmente para poder comprobar la efectividad de este sistema de control, se decidió montar el vehículo de forma que las ruedas traseras pudiesen girar pero sin estar en contacto con el suelo. Esto nos permitiría utilizar el Plotter monitor de Arduino para comprobar la efectividad del controlador PI diseñado.

En la siguiente imagen puede apreciarse el montaje realizado que nos permite conectar la placa de Arduino al ordenador sin que el vehículo se desplace.

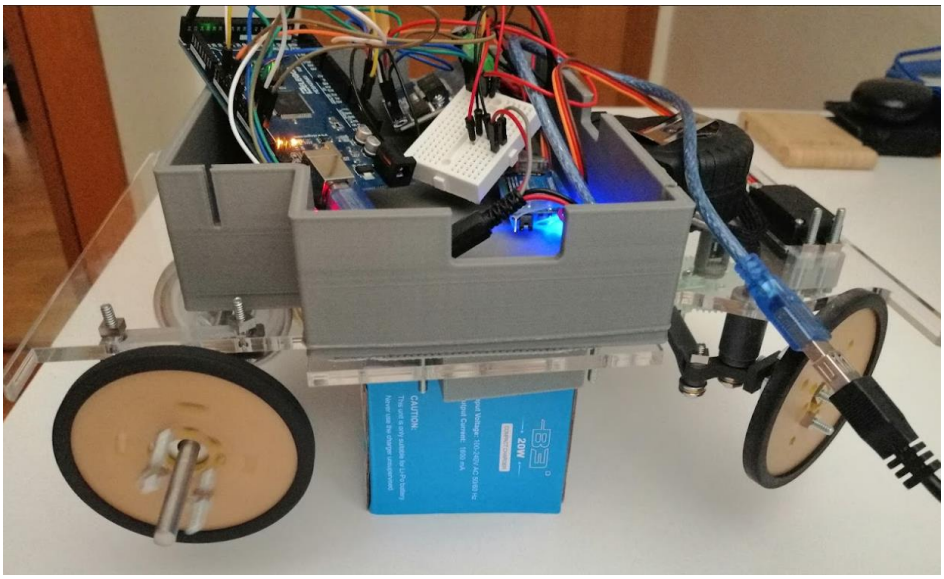


Ilustración 181: Montaje prueba PI

Obviamente el controlador PI calculado de esta forma no será perfecto ya que no se está aplicando ninguna fuerza que se oponga al giro del motor salvo al rozamiento de los componentes del sistema de tracción.

La velocidad de referencia utilizada oscila entre 30 y -30 cm/s ya que está es la velocidad máxima que permite nuestro vehículo.

Inicialmente se probó a controlar la velocidad del vehículo con un controlador P, con un valor de Kp = 10. En la gráfica inferior puede apreciarse los resultados de dicha prueba:

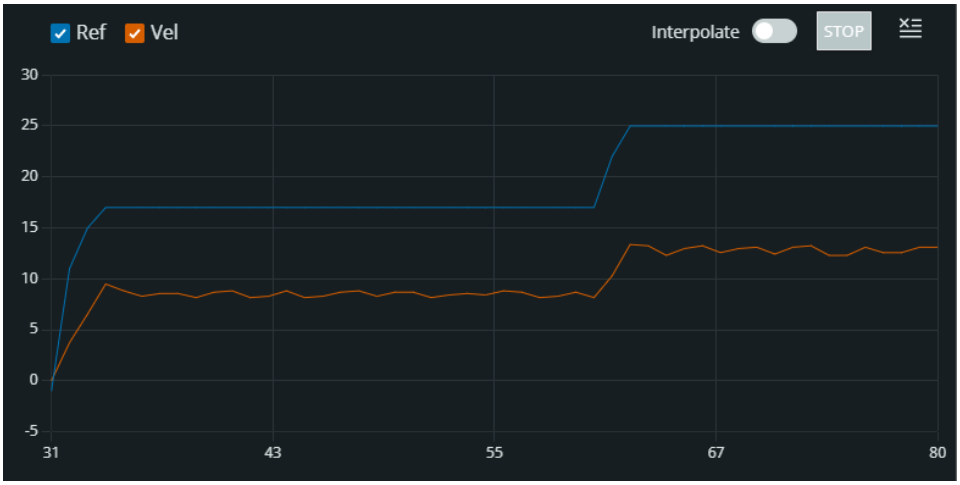


Ilustración 182: Prueba velocidad con P

Como puede apreciarse la 'velocidad' del vehículo queda lejos de la deseada, con lo que puede concluirse que dicho controlador será insuficiente para controlar la velocidad del vehículo.

A continuación, veremos la velocidad obtenida para el caso en el que se implemente un controlador PI con valores de Kp = 10 y KI = 5:

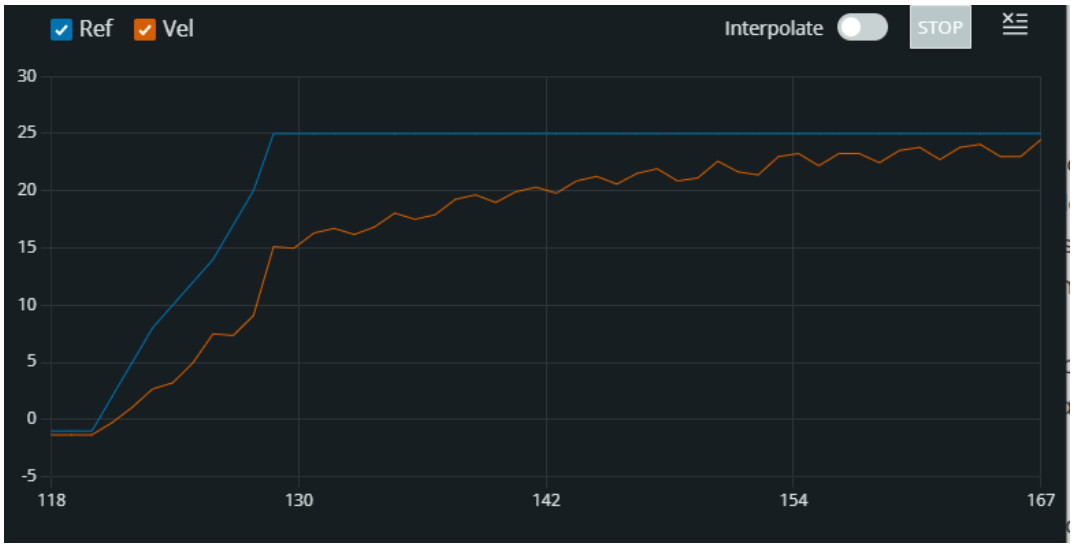


Ilustración 183: Prueba velocidad con PI -1

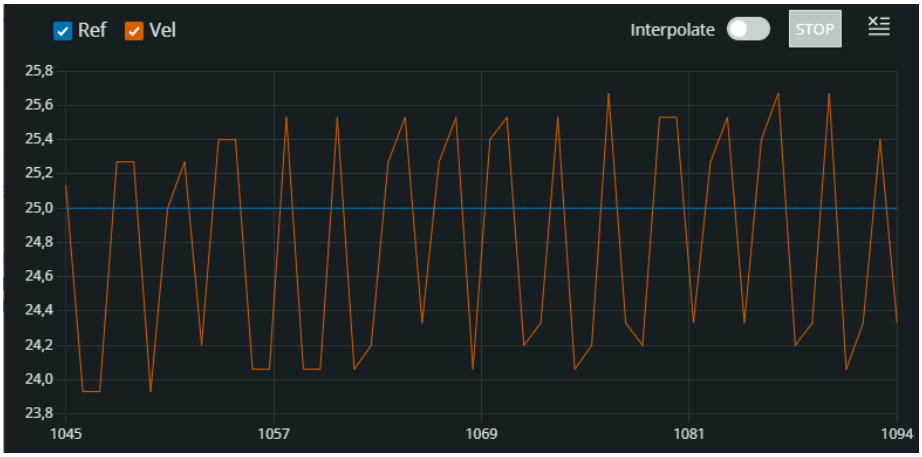


Ilustración 184: Prueba velocidad con PI - 2

Como podemos comprobar, la introducción del valor integral en el controlador permite que el vehículo sea capaz de alcanzar la velocidad deseada en un tiempo razonable.

Tras haber comprobado la efectividad del controlador sin que se desplace el vehículo, procedimos a reprogramar la aplicación que utilizábamos en la aplicación de Bluetooth Electronics así como el código de Arduino de forma que se pudiese enviar mensajes del microcontrolador a la aplicación móvil. Esto nos ha permitido comprobar la efectividad del controlador mediante lo siguiente:

- Representación visual de la velocidad del vehículo así como de su referencia
- Posibilidad de variar los valores de Kp y KI sin tener que variar el propio código fuente del microcontrolador.

La nueva interfaz diseñada tiene el siguiente aspecto:

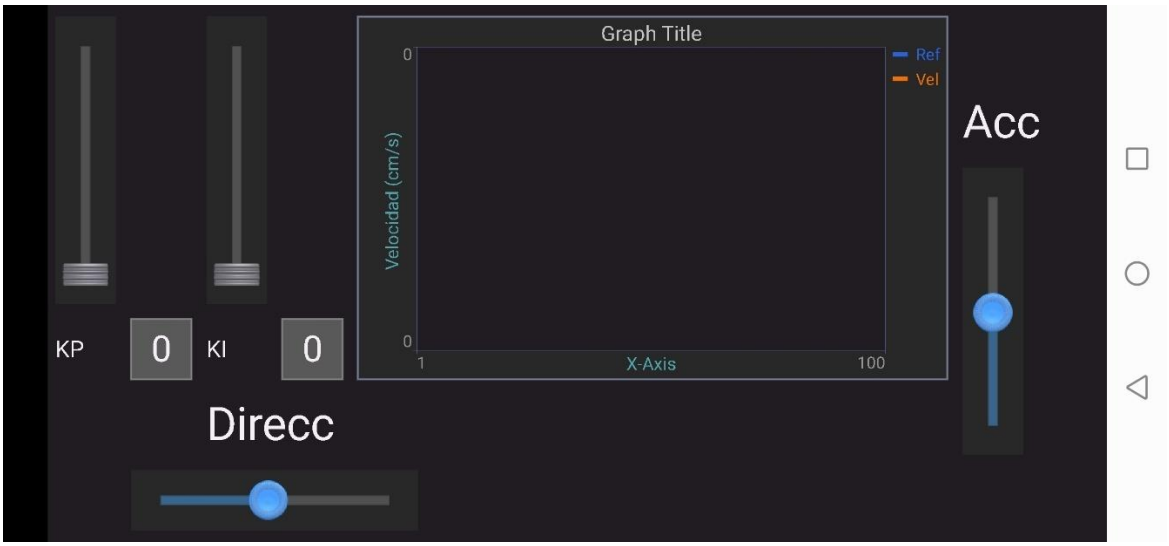


Ilustración 185: Interfaz Control PI

Esta dispone de 2 deslizadores verticales en su parte izquierda para variar y visualizar los valores de KP y KI del controlador PI, un slider horizontal que controla el giro de las ruedas directrices, un slider vertical que nos permite variar la referencia de velocidad que le enviamos al microcontrolador y finalmente una gráfica que nos permite observar en tiempo real la velocidad del vehículo, así como la referencia enviada.

En las siguientes imágenes puede apreciarse el efecto de variar el valor de KP durante el funcionamiento del vehículo:

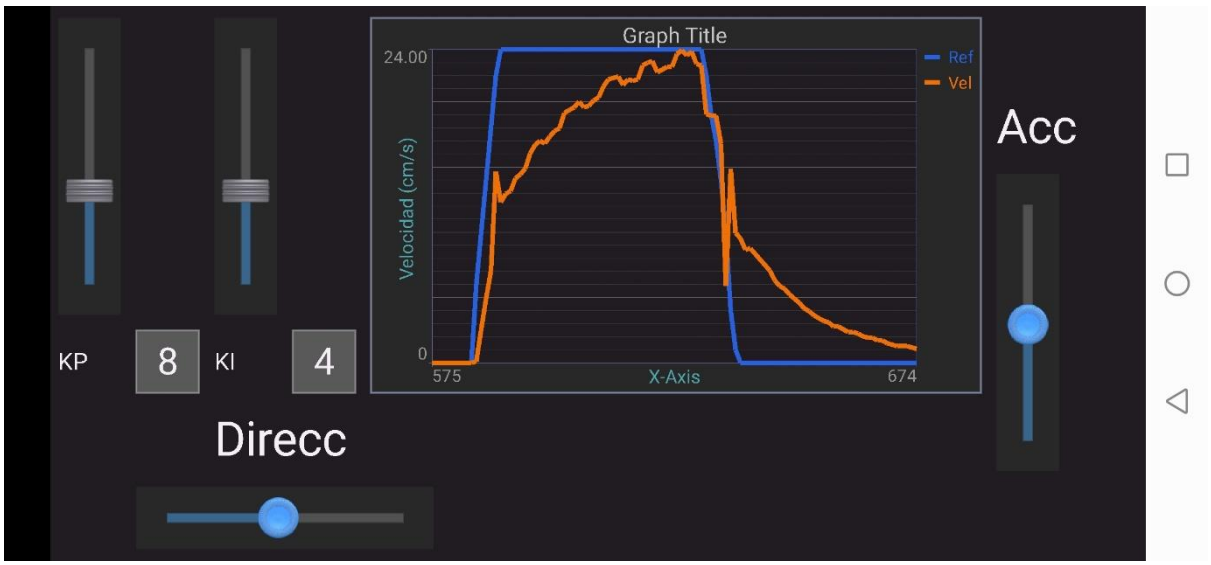


Ilustración 186: Control PI - KP = 8

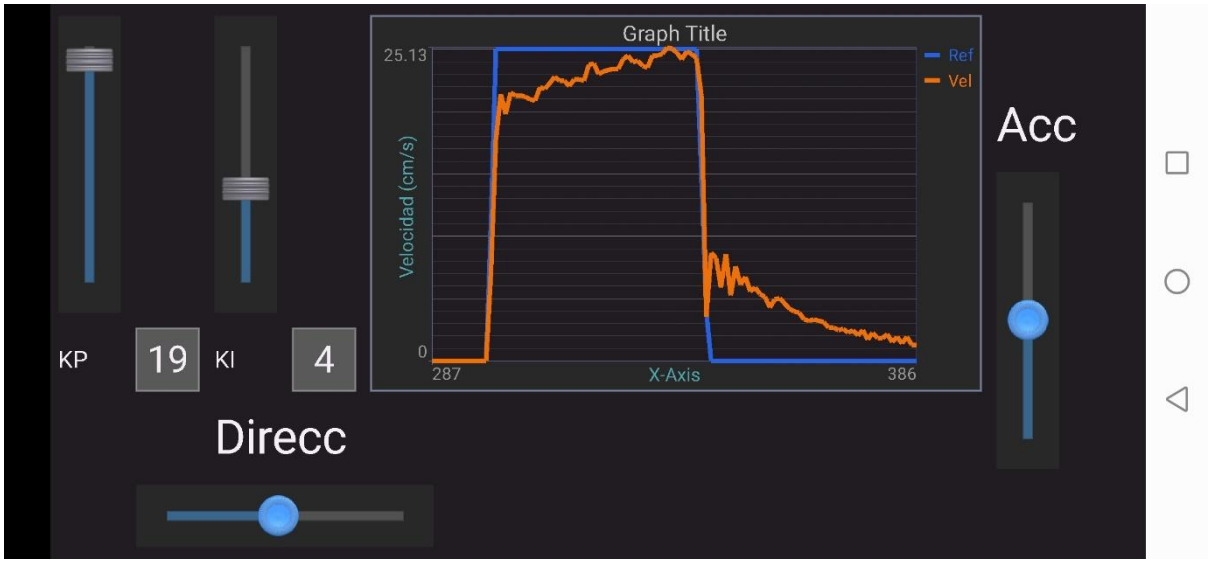


Ilustración 187: Control PI - KP = 19

Como es de esperar, al aumentar el valor de Kp del controlador, se reduce el tiempo que tarda el vehículo en alcanzar la velocidad deseada.

4.5 Problemas encontrados

En este apartado vamos a describir algunos de los problemas encontrados a la hora de realizar el montaje del vehículo, así como la solución que se decidió aportar para solucionar cada problema

Elevados tiempos de entrega

La mayoría de las componentes que se pidieron online o encargaron a la universidad tuvieron un tiempo de entrega de alrededor de una semana, lo cual no supuso un problema mayor.

Sin embargo el pedido que contenía el mayor número de componentes mecánicos tuvo problemas con la disponibilidad de algunos componentes. Se procedío a contactar con la empresa suministradora, la cual aclaró que en pocos días estarían todos los componentes listos para el envío.

Sin embargo, si que supuso un tiempo de espera de alrededor de un mes, tiempo el cual por suerte se pudo aprovechar para redactar la siguiente memoria.

Envío de pieza defectuosa

Durante el transcurso del montaje del sistema de tracción, resulta ser que el piñón que va montado sobre el eje de salida del motor principal dispone de un tornillo prisionero para poder montarse adecuadamente al eje del motor como podemos ver en la imagen:

Ilustración 188: Tornillo prisionero piñón

El componente enviado por la suministradora contenía un tornillo defectuoso, ya que no disponía de huella para poder enroscar dicha pieza sobre el piñón. Esto se puede apreciar en la siguiente imagen:

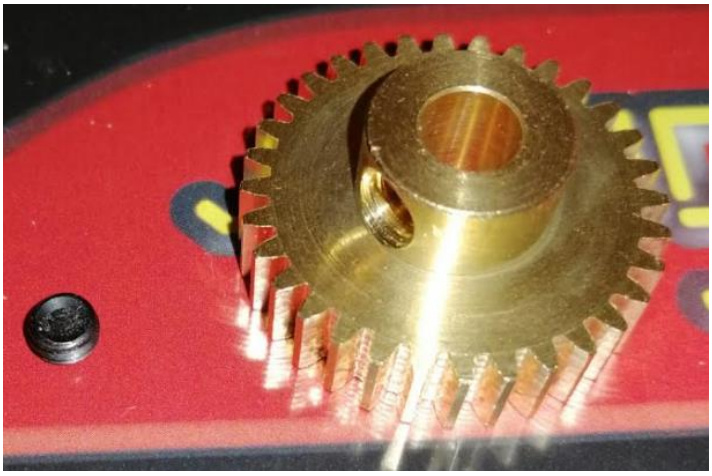


Ilustración 189: Problema piñón

La propia tienda que suministro este producto ofreció la posibilidad de enviar una nueva pieza que no fuese defectuosa. Sin embargo debido a los elevados tiempos de entrega causados por esta suministradora, se optó por buscar una solución alternativa.

Inicialmente se trató de encontrar un tornillo prisionero que pudiese enroscar sobre el piñon. Esto resulto en un fracaso ya que el tipo de rosca que utiliza el piñón es del estándar ANSI con lo que no hubo posibilidad de encontrar un tornillo adecuado sin aumentar excesivamente el presupuesto del vehículo.

La solución adoptada fue llevar la pieza en cuestión al taller de la universidad, mecanizar una rosca métrica en el agujero que engrana con el tornillo y simplemente usar un tornillo prisionero de métrica M5 para el montaje.

Desconexión del módulo Bluetooth

Este problema ya se mencionó en el apartado del montaje eléctrico, en el que el módulo Bluetooth se desconectaba brevemente perdiendo la conexión con el smartphone. Esto resultó ser debido a la elevada corriente exigida por el Servomotor que controla el sistema de dirección.

Para solventarlo se conectaron ambos motores a la batería de 12V, la cual es capaz de aportar suficiente potencia eléctrica como para alimentar ambos motores. Para reducir la tensión de la batería a una tensión adecuada para el Servomotor nos aprovechamos del regulador de tensión que se aprecia en la siguiente imagen:

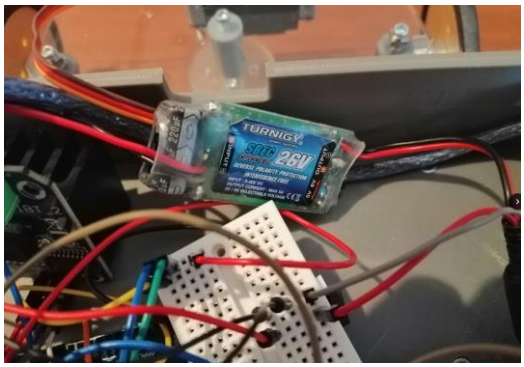


Ilustración 190: Regulador de tensión

Pieza partida

Tras el viaje realizado a la universidad para resolver el último problema que se mencionó, la bieleta izquierda del vehículo se partió en 2. Esto se puede apreciar en la siguiente imagen:

Ilustración 191: Pieza rota

La solución a esto fue simplemente pedir de nuevo la pieza al departamento de impresión 3D de la universidad.

Sin embargo un problema adicional que surgió fue que dicha pieza tenía montada sobre la misma un rodamiento el cual se quedó completamente incrustado en la pieza, realizando imposible su desmontaje mediante los medios que disponía el alumno.

Ilustración 192: Bieleta incrustada en el rodamiento

Finalmente para su desmontaje se optó por llevar la pieza a una ferretería cercana, en la que fueron capaces de separar ambas piezas.

5 presupuesto

En este capítulo se incluye el presupuesto final que supondría la realización completa de este proyecto. Se incluye el importe de todos los componentes necesarios para el montaje del vehículo tanto a nivel de material y hardware utilizado como del coste personal que implicaría el diseño, simulación y montaje del vehículo.

Conviene mencionar que las cifras aquí mostradas difieren del coste real del prototipo montado por el estudiante ya que las impresiones 3D y el corte del metacrilato corren por cargo de la universidad, entre otros conceptos.

5.1 Coste materiales

A continuación, veremos el coste relativo al material y hardware utilizados en el montaje del prototipo diseñado. Ya que el proyecto se ha realizado por completo por un estudiante, se incluyen en el listado los proveedores de los distintos artículos adquiridos.

Cabe mencionar que todos los precios que se encuentran en esta lista incluyen el coste del IVA.

Concepto	Proveedor	Marca	Ud.	Precio/Ud (€)	Total (€)
Motor 12 V con encoder	Bricogeek	Pololu	1	48.40	48.40
Eje tipo D 12"x1/4"	Robotshop	Actobotics	1	5.48	5.48
Soporte motor metal	BricoGeek	Pololu	1	8.40	8.40
Par de ruedas 3"	Robotshop	Actobotics	2	4.77	9.54
Montaje de centro de tornillo 0.77"-1/4"	Robotshop	Actobotics	3	5.87	17.61
Actobotics Piñon 32D-6mm	Robotshop	Actobotics	1	8.95	8.95
Actobotics Engranaje 64T-0.5"	Robotshop	Actobotics	1	15.28	15.28
Rodamientos bolas 1/4"-1/2" (x2)	Robotshop	Actobotics	1	2.46	2.46
Driver motor 12 V	Amazon	KYYKA	1	8.99	8.99
Batería 12V	Bricogeek	Bricogeek	1	24.20	24.20
Cargador batería Litio	Amazon	ICQUANZX	1	12.00	12.00
Buje de rodamientos (par)	Robotshop	Actobotics	1	8.23	8.23
Rodamientos axiales F5-10M (x10)	Amazon	Jectse	1	11.00	11.00
Servomotor MG996R	Amazon	AZDelivery	1	10.99	10.99
Tornillos 3/4" 6-32 (x25)	Robotshop	Actobotics	1	3.35	3.35
Tuercas 6-32 (x24)	Robotshop	Actobotics	2	5.87	11.74
Tornillos 3/4" 6-32 (x25)	Robotshop	Actobotics	1	4.62	4.62
Arduino Mega 2560	Amazon	ELEGOO	1	25.00	25.00
Módulo Bluetooth HC-05	Amazon	ARCELI	1	7.49	7.49
Power Bank 1000 mAh	PcComponentes	Xiaomi	1	30.10	30.10
Cables (x40)	Amazon	ELEGOO	1	4.50	4.50
PLA piezas impresas	UPV	-	1	5.00	4.00
Metacrilato 50x50 cm	UPV	-	1	14.00	14.00
Costes envío Amazon	Amazon	Amazon	1	3.78	3.78
Costes envío Robotshop	Robotshop	Robotshop	1	16.29	16.29
Costes envío Bricogeek	Bricogeek	Bricogeek	1	4.78	4.78
				Total:	321.18 €

Por tanto el coste total de los materiales asciende a 321.18 €.

5.2 Coste Personal

A la hora de obtener el coste del personal se contabilizará que el estudiante haya dedicado una media de 60 horas mensuales durante un período de 6 meses, desglosado en las distintas tareas realizadas para el desarrollo del proyecto. Tomando como base el XIX Colectivo nacional de empresas de Ingeniería y oficinas de estudios técnicos[8], situaremos el coste del ingeniero en 21.85€/h.

Se incluye además el tiempo requerido por el técnico de laboratorio en imprimir las piezas 3D así como de obtener las piezas cortadas por láser de metacrilato. El coste de dicho técnico lo situaremos en 17.48€/h.

Concepto	**Coste (€/h)**	**Horas totales**	**Total (€)**
Definición de concepto	21.85	6	131.10
Investigación relativa al proyecto	21.85	40	874.00
Diseño 3D simulación	21.85	30	655.50
Simulación	21.85	180	3933.00
Diseño prototipo	21.85	25	546.25
Montaje prototipo	21.85	4	87.40
Programacion	21.85	10	218.50
Prueba y arreglo errores	21.85	20	437.00
Impresión piezas	17.48	1.5	26.22
Corte laser metacrilato	17.48	1	17.48
Otros	21.85	40	874.00
		Total	7,800.45 €

Con ello el coste relativo al personal involucrado en el proyecto asciende a 7,800.45 €.

5.3 Coste Licencias

Debido a la utilización de programas informáticos cuyo uso requieren de una licencia profesional, se ha incluido el coste de dichas licencias. Se considera que cualquier licencia informática necesaria ha sido utilizada por un período de 7 meses, con lo que se contabilizará el coste correspondiente a disponer de una licencia durante ese período de tiempo.

Programa	Desarrollador	Coste anual	Coste 7 meses
MATLAB & Simulink	MathWorks	800.00 €	466.67 €
SolidWorks	Dassault Systèmes	2,557.00 €	1,491.58 €
		Total	1,958.25

El coste relacionado a las licencias necesarias para realizar este proyecto ascendería a 1,958.25€.

5.4 Coste total

Tras haber desglosado el importe de los distintos conceptos del proyecto, contabilizaremos el coste total de realizar el proyecto de diseño e implementación de un vehículo de Ackermann.

Concepto	Precio
Coste material	321.18 €
Coste personal	7,800.45 €
Coste licencias	1,958.25 €
Total:	**10,079.88 €**

Con ello el coste total del proyecto quedaría fijado en 10,079.88 €.

6 Pliego de condiciones

Este pliego de condiciones actúa a modo de extensión del contrato entre propiedad y contratista para la ejecución del proyecto de diseño de Vehículo de Ackermann.

6.1 Objeto del pliego

En este capítulo se pretenden estipular las condiciones técnicas necesarias a cumplir para la realización de un vehículo de Ackermann. Debido a que la memoria trata de un trabajo académico, el pliego no contendrá extensas condiciones detalladas sobre la implementación del trabajo en un ambiente industrial.

6.2 Condiciones generales

La normativa aplicable en términos legales vendrá definida por el Boletín oficial de la Universitat Politécnica de Valencia en relación a la redacción de proyectos de Trabajos de finde grado y máster.

Durante la realización del proyecto se deberá respetar en todo momento la normativa del Reglamento de Baja Tensión.

Los materiales escogidos para el montaje del vehículo deberán ser tales de forma que permitan el adecuado funcionamiento del mismo así como se describe en la memoria.

6.3 Condiciones ejecución

Para la adecuada realización de este proyecto se deben seguir unas pautas claras que solo podrá realizar personal con la formación adecuada que se considere necesaria para la realización de cada tarea descrita a continuación.

Los pasos a seguir para la realización de un vehículo de Ackerman son las siguientes:

1. Diseño conceptual del vehículo con un programa de diseño en CAD
2. Importación de dicho vehículo a un programa de Simulación
3. Simulación del vehículo diseñado en las condiciones descritas en el documento
4. Rediseño del vehículo en base a los resultados obtenidos en Simulación
5. Diseño de un prototipo utilizando piezas comerciales y/o de fabricación propia
6. Montaje del vehículo diseñado
7. Programación del sistema de control manual del vehículo
8. Prueba inicial del vehículo en un entorno controlado
9. Ajuste de errores encontrado en la prueba inicial

6.4 Pruebas y ajustes finales

Tras realizar el montaje del vehículo se procederá a realizar cualquier ajuste sobre el mismo que asegure el adecuado funcionamiento para el que se diseñó el vehículo.

7 Conclusiones – Trabajos futuros

El trabajo realizado muestra el trabajo realizado para el diseño del prototipo de un vehículo de Ackermann. Con este trabajo culmina la etapa como estudiante del alumno tras varios años de esfuerzo y dedicación.

El diseño del siguiente vehículo ha supuesto una gran profundización en gran parte de los conceptos estudiados en el Máster tal y como el diseño de Sistemas de control, funcionamiento de componentes de electrónica y programación.

El diseño del vehículo mediante el programa de CAD SolidWorks ha permitido realizar un diseño apto para simulación que nos permita comprobar el funcionamiento del mismo. Nos ha permitido además comprobar como se podrían implementar en el vehículo componentes comerciales de distintas marcas.

La simulación del vehículo nos ha permitido comprobar el funcionamiento del propio vehículo antes de construir el mismo así como de comprobar el adecuado funcionamiento del sistema de seguimiento de trayectoria que se ha diseñado para el vehículo.

A nivel de objetivos se puede considerar que el trabajo realizado logra cumplir con las expectativas de simulación, sin embargo, el prototipo diseñado requiere de una mejora en el diseño del mecanismo de dirección para poder ser capaz de utilizarse para realizar un seguimiento de trayectoria.

El trabajo ha resultado de gran interés debido a ser un montaje real hecho desde 0 por el propio alumno, esto ha supuesto una gran libertad a la hora de diseñar el vehículo, así como un gran reto en lo que supone el diseño de algo real.

Algunos de los aspectos que se podrían mejorar de cara a un futuro son los siguientes:

- En simulación, en vez de aplicar un par directamente sobre el eje, implementar un motor de CC, al que se le aplique una tensión controlada
- Mejora del diseño del sistema de dirección del vehículo
- Implementación del sistema de control de trayectoria una vez mejorado el sistema de dirección del vehículo

Es de la opinión del alumno que este trabajo podría ser de gran utilidad en un futuro en el que se pretenda profundizar en el diseño de un sistema de seguimiento de trayectorias para un vehículo autónomo.

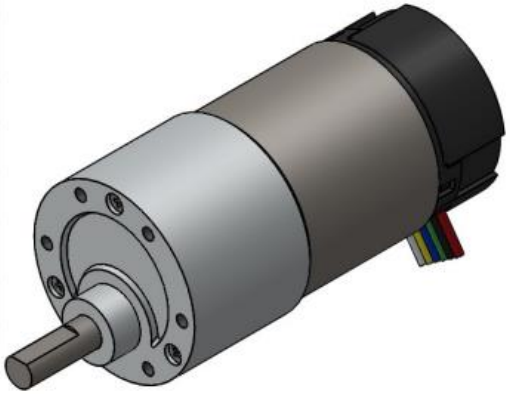
Anexo I: Planos

Contenido:

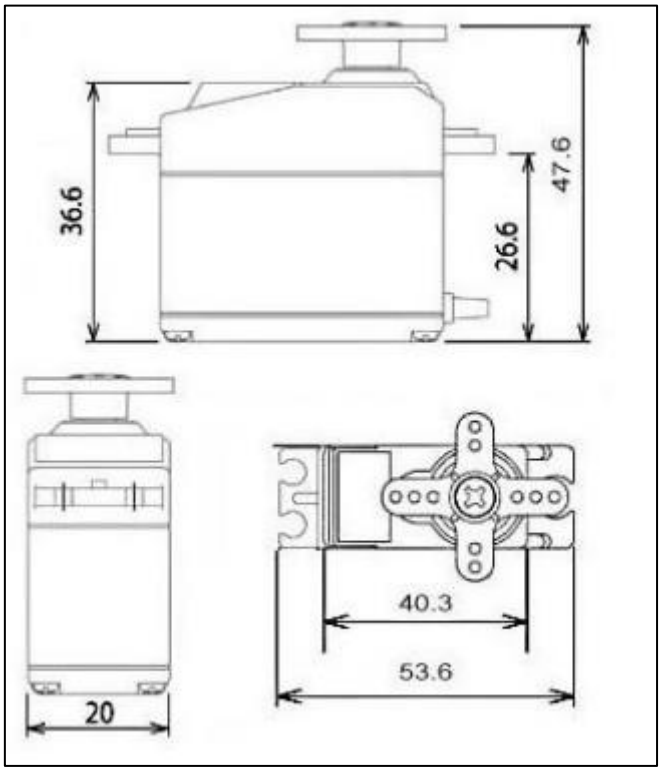
- Motor 12 V con encoder 70:1 Pololu
- Servomotor MG996R
- Rueda de precisión 3” - Actobotics
- Tuercas cuadradas 6-32
- Eje tipo D de 12” - ¼” - Actobotics
- Base
- Soporte eje
- Barra dirección
- Bieleta de dirección
- Brazo de dirección
- Soporte bieletas
- Carcasa electrónica
- Vehículo General
- Vehículo Dirección
- Vehículo Tracción

Motor 12 V con encoder 70:1 Pololu

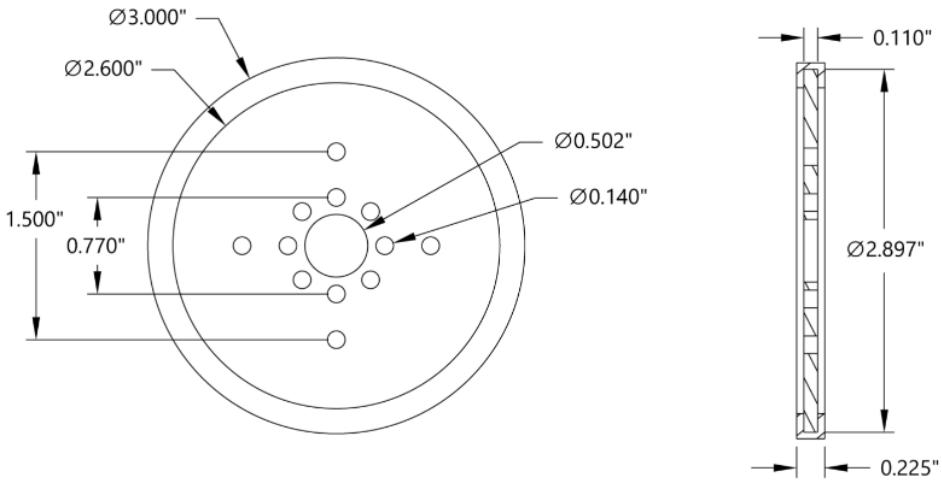
Gear ratio	L
19:1	22 mm [0.87 in]
30:1	22 mm [0.87 in]
50:1	24 mm [0.94 in]
70:1	24 mm [0.94 in]
100:1	26.5 mm [1.04 in]
131:1	26.5 mm [1.04 in]
150:1	26.5 mm [1.04 in]



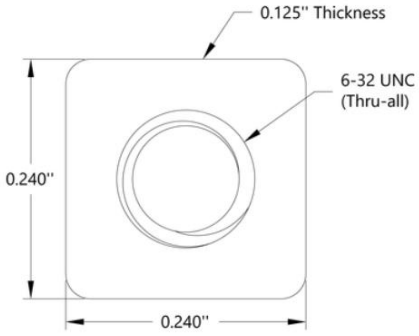
Servomotor MG996R



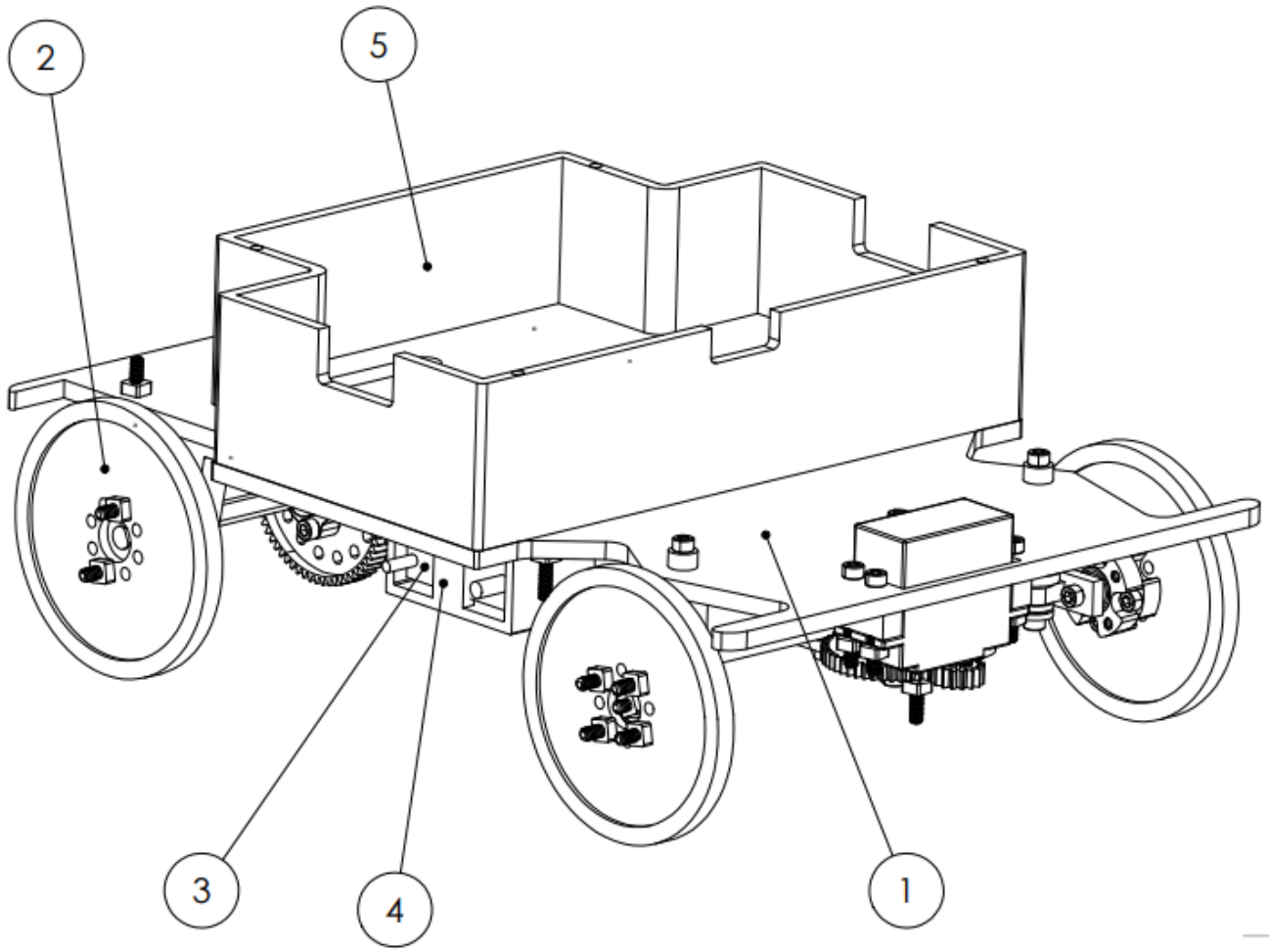
Rueda de precisión 3” - Actobotics



Tuercas cuadradas



Eje tipo D de 12” - ¼” - Actobotics



Nº	Descripción	Cantidad
1	Base	1
2	Rueda	4
3	Batería 12 V	1
4	Soporte batería 12 V	2
5	Carcasa electrónica	1

UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:

Drawing made in
Solidworks 3D

DEBURR AND BREAK SHARP EDGES

DO NOT SCALE DRAWING

DATE: 01/07/2022

Master Ingeniería Mecatrónica

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

TITLE:
Vehículo completo

Project:
Vehículo de Ackermann

MATERIAL:

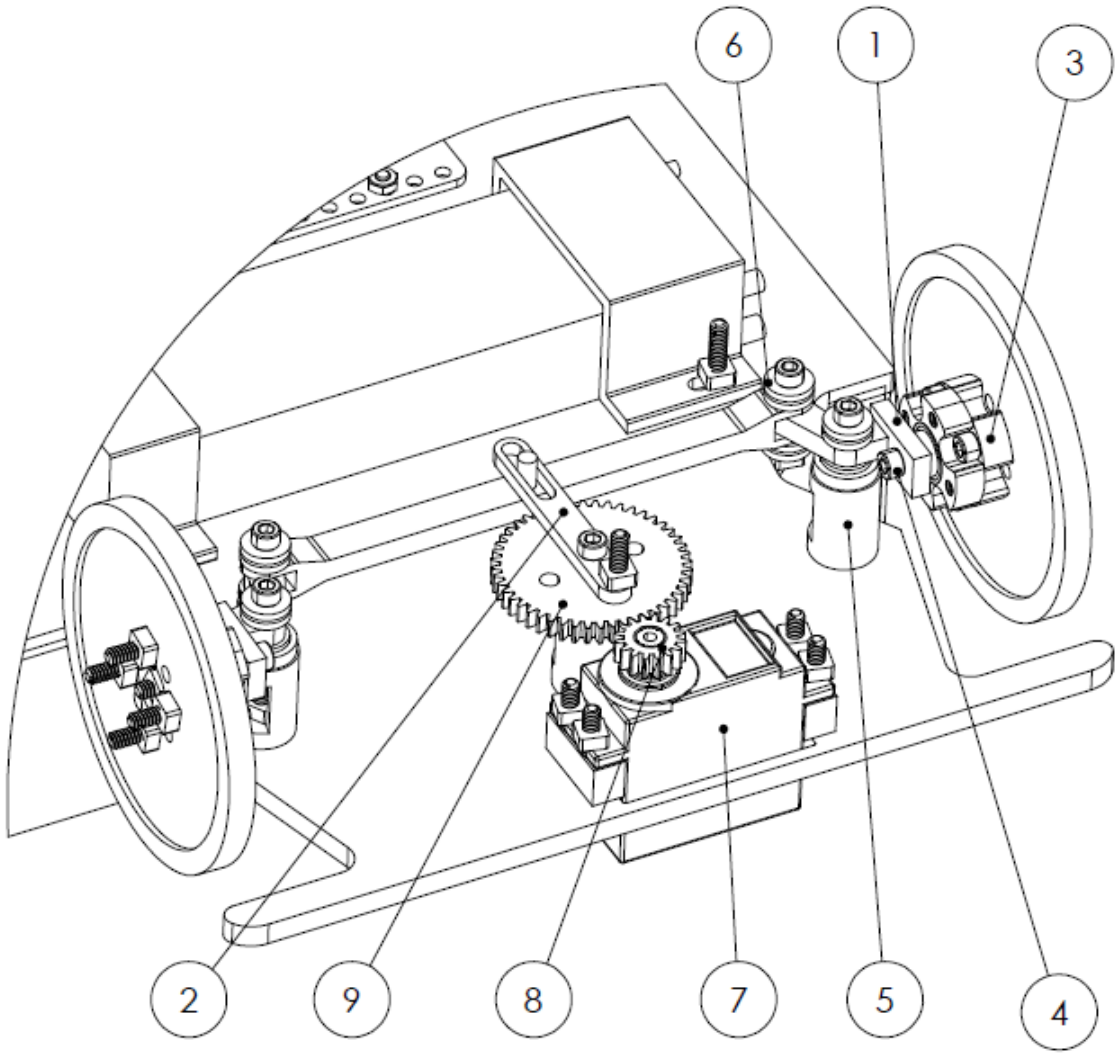
WEIGHT:

Author
Sindre Johannesen

A4

SCALE:1:5

SHEET 1 OF 1



N°	Descripción	Cantidad
1	Bieleta	2
2	Brazo dirección	1
3	Sujecion ruedas	2
4	Tornillo 6-32 1-1/2"	2
5	Soporte bieletas	2
6	Rodamiento Axial	9
7	Servomotor	1
8	Engrane 16D metacrilato	1
9	Engrane 64D metacrilato	1

UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:

Drawing made in
Solidworks 3D

DEBURR AND BREAK SHARP EDGES

DO NOT SCALE DRAWING

DATE: 01/07/2022

Master Ingeniería Mecatrónica

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

TITLE:
Vehículo completo - Dirección

Project:
Vehículo de Ackermann

MATERIAL:

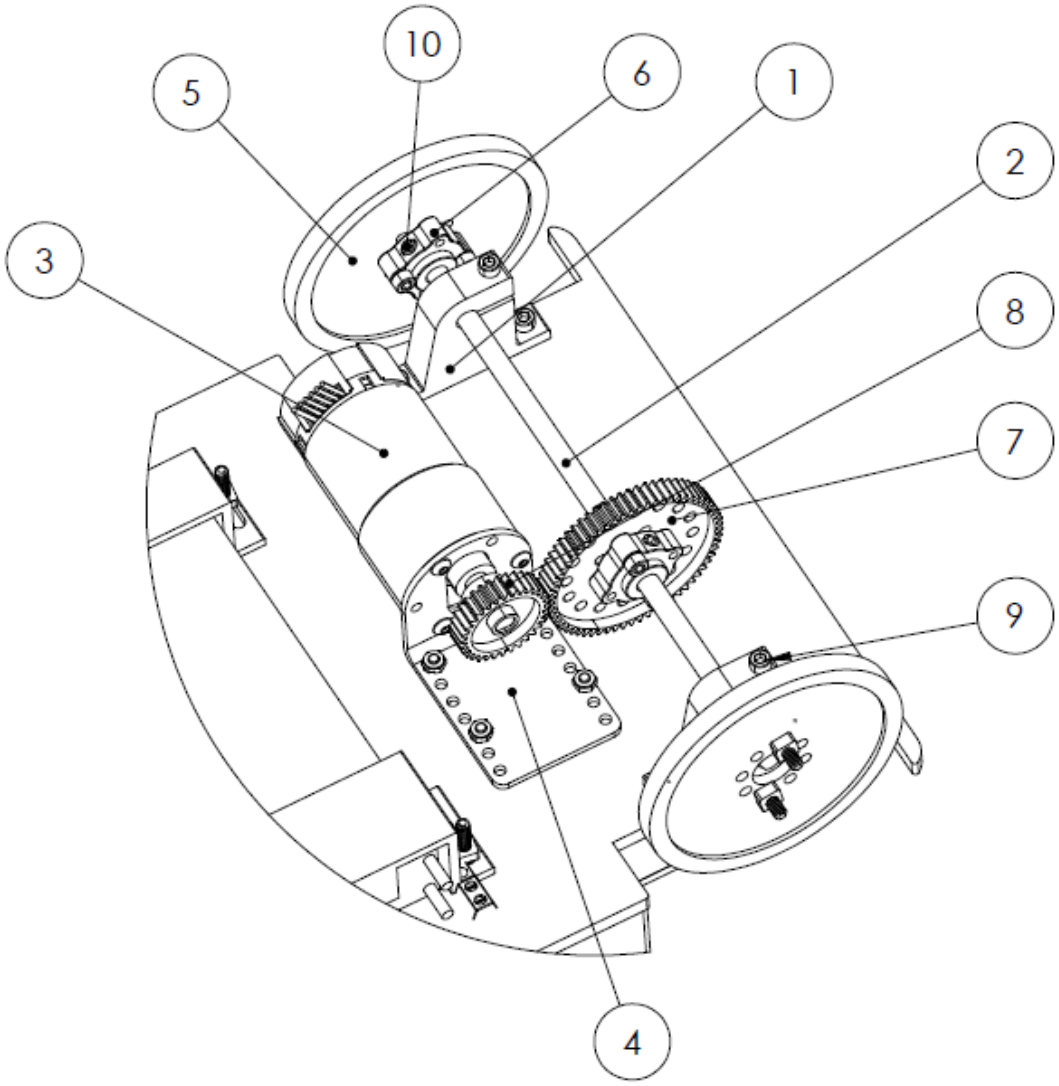
Author
Sindre Johannesen

A4

WEIGHT:

SCALE:2:3

SHEET 1 OF 1



N°	Descripción	Cantidad
1	Soporte Eje	2
2	Eje	1
3	Motor 12 V	1
4	Soporte motor	1
5	Ruedas	4
6	Sujecion ruedas	3
7	Engrane 64D	1
8	Engrane 32 D	1
9	Tornillo 6-32 3/4"	29
10	Tornillo prisionero	4

UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:

Drawing made in
Solidworks 3D

DEBURR AND BREAK SHARP EDGES

DO NOT SCALE DRAWING

DATE: 01/07/2022

Master Ingeniería Mecatrónica

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

TITLE:
Vehículo completo - Tracción

Project:
Vehículo de Ackermann

MATERIAL:

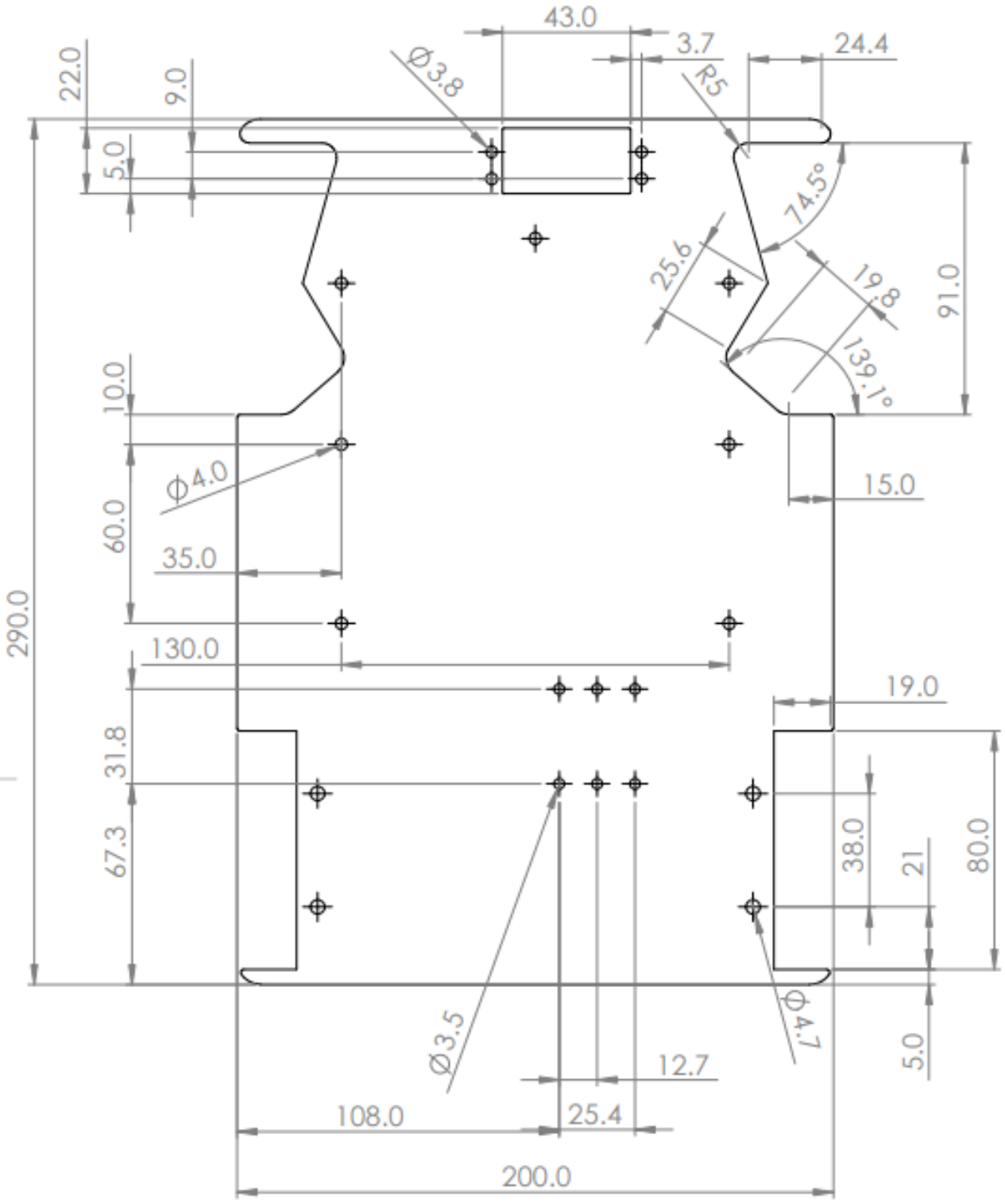
WEIGHT:

Author
Sindre Johannesen

A4

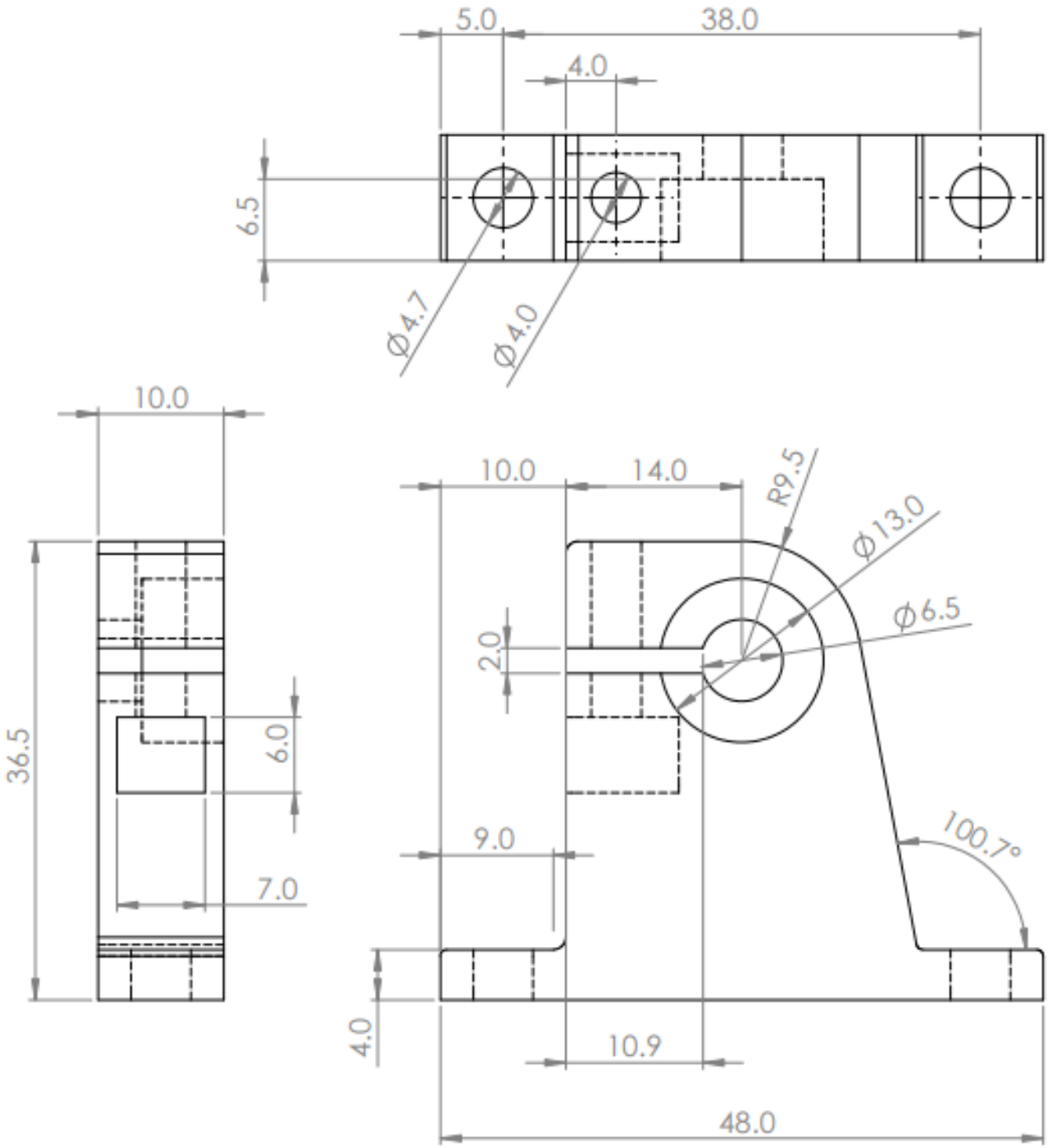
SCALE:1:5

SHEET 1 OF 1

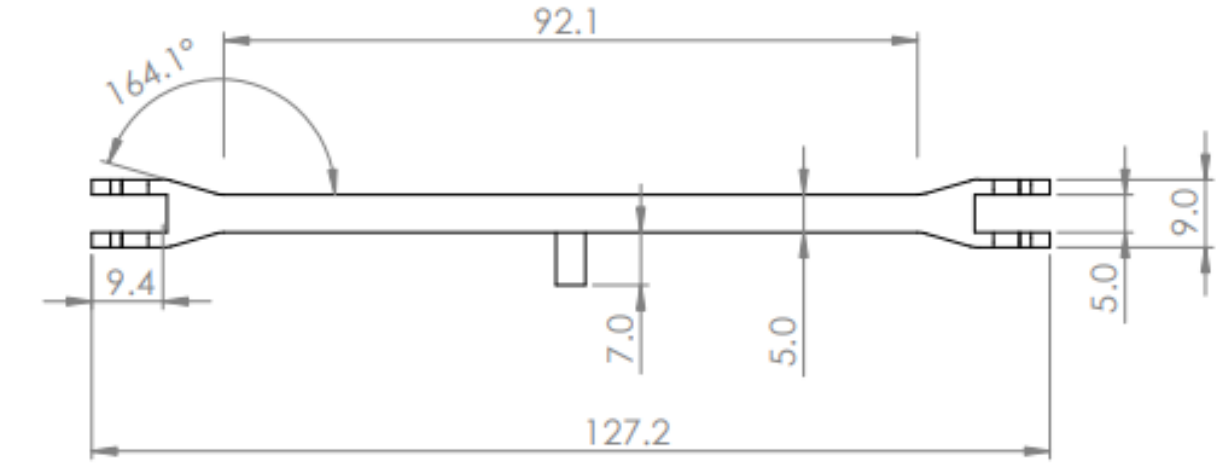


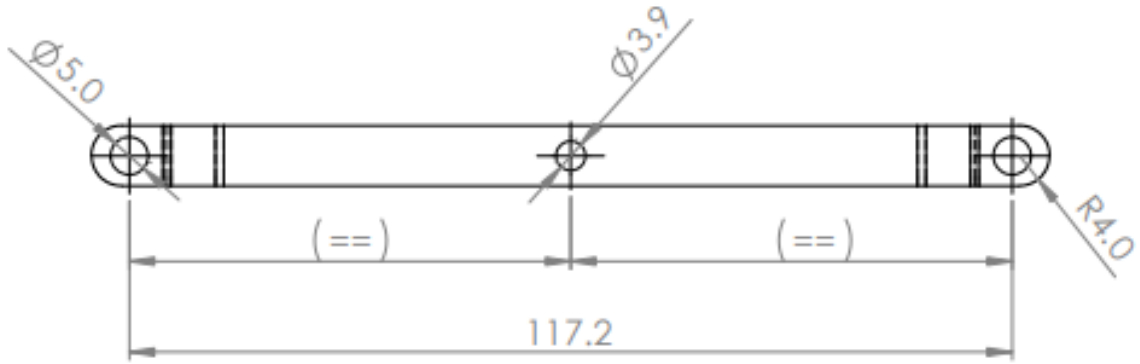
La pieza tiene un espesor de 5 mm

UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:	Drawing made in Solidworks 3D	DEBURR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING	DATE: 01/07/2022
			Master Ingeniería Mecatrónica	
UNIVERSITAT POLITÈCNICA DE VALÈNCIA			TITLE: Base Vehículo	
Vehículo de Ackermann	MATERIAL: Metacrilato		Author: Sindre Johannesen	A4
	WEIGHT:		SCALE:1:2	SHEET 1 OF 1

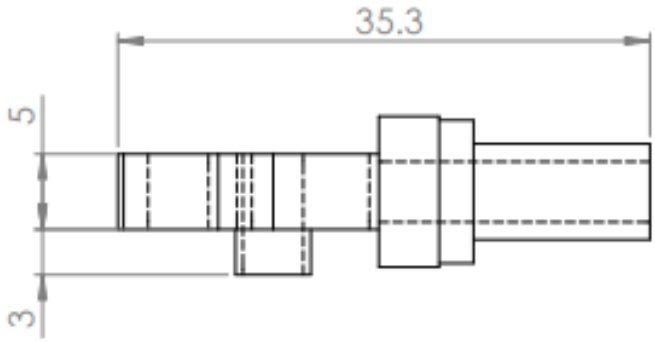


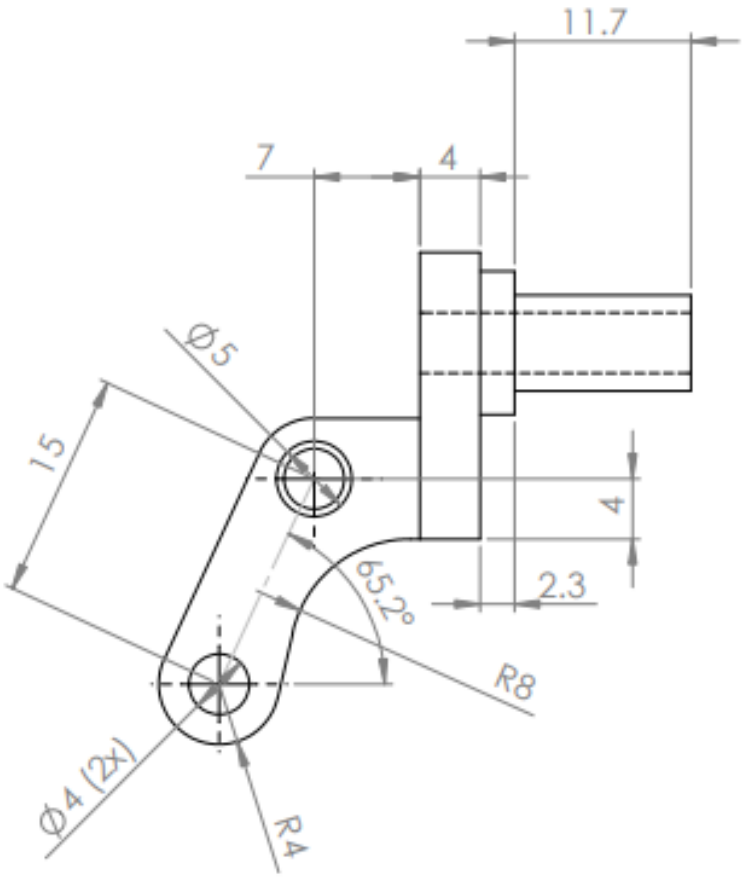
UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:	Drawing made in Solidworks 3D	DEBURR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING	DATE: 01/07/2022
			Master Ingeniería Mecatrónica	
UNIVERSITAT POLITÈCNICA DE VALÈNCIA			TITLE: Soporte EJE	
Project: Vehículo de Ackermann	MATERIAL: PLA		Author Sindre Johannesen	A4
	WEIGHT:		SCALE:1:1	SHEET 1 OF 1

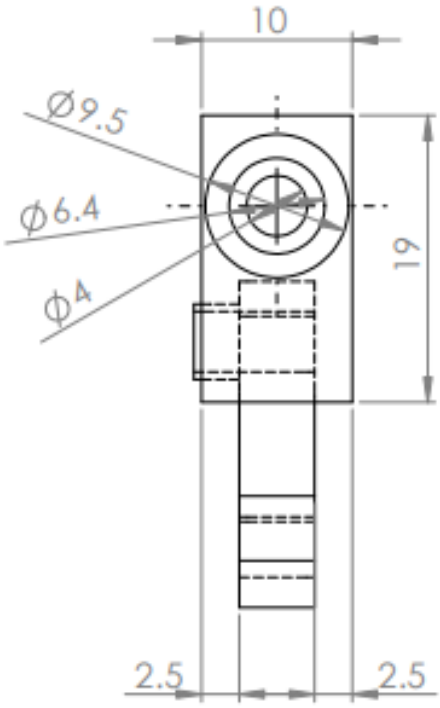




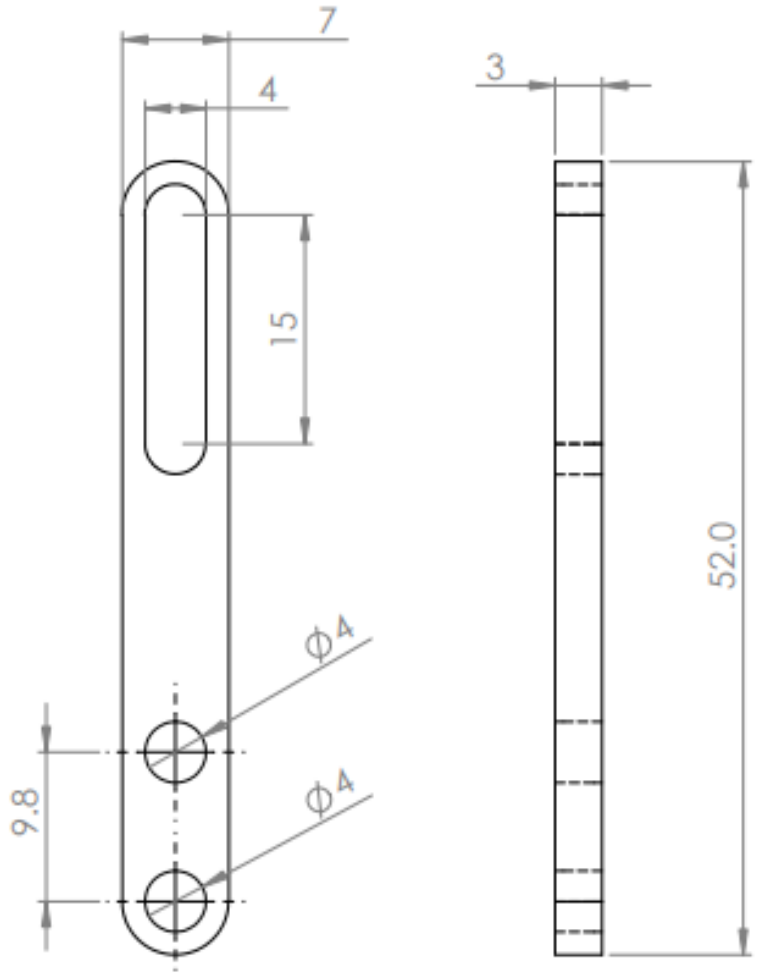
UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:	Drawing made in Solidworks 3D	DEBURR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING	DATE: 01/07/2022
			Master Ingeniería Mecatrónica	
UNIVERSITAT POLITÈCNICA DE VALÈNCIA			TITLE: Barra dirección	
Project: Vehículo de Ackermann	MATERIAL: PLA		Author Sindre Johannesen	A4
	WEIGHT:		SCALE:1:2	SHEET 1 OF 1



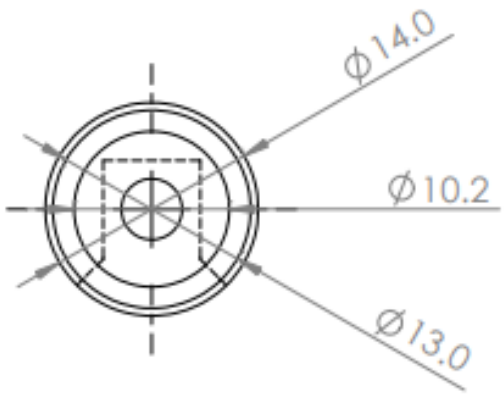


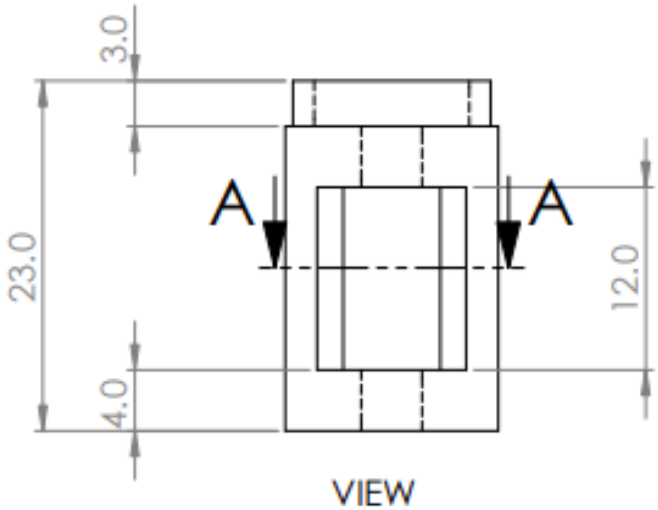


UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:	Drawing made in Solidworks 3D	DEBURR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING	DATE: 01/07/2022
			Master Ingeniería Mecatrónica	
UNIVERSITAT POLITÈCNICA DE VALÈNCIA			TITLE: Bieleta dirección	
Project: Vehículo de Ackermann	MATERIAL: PLA		Author Sindre Johannesen	A4
	WEIGHT:		SCALE:2:1	SHEET 1 OF 1

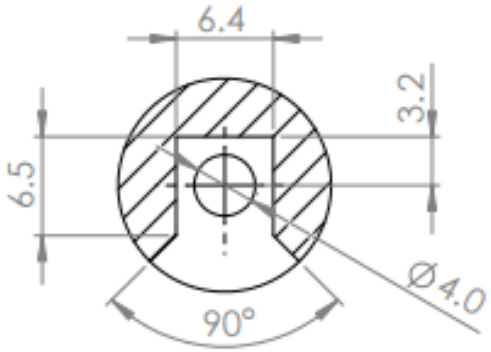


UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:	Drawing made in Solidworks 3D	DEBURR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING	DATE: 01/07/2022
			Master Ingeniería Mecatrónica	
UNIVERSITAT POLITÈCNICA DE VALÈNCIA			TITLE: Brazo Direcccion	
Project: Vehículo de Ackermann	MATERIAL: Metacrilato		Author Sindre Johannesen	A4
	WEIGHT:		SCALE:2:1	SHEET 1 OF 1



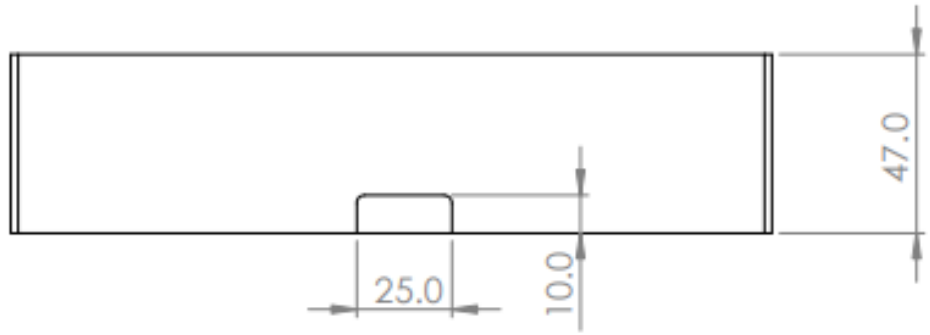


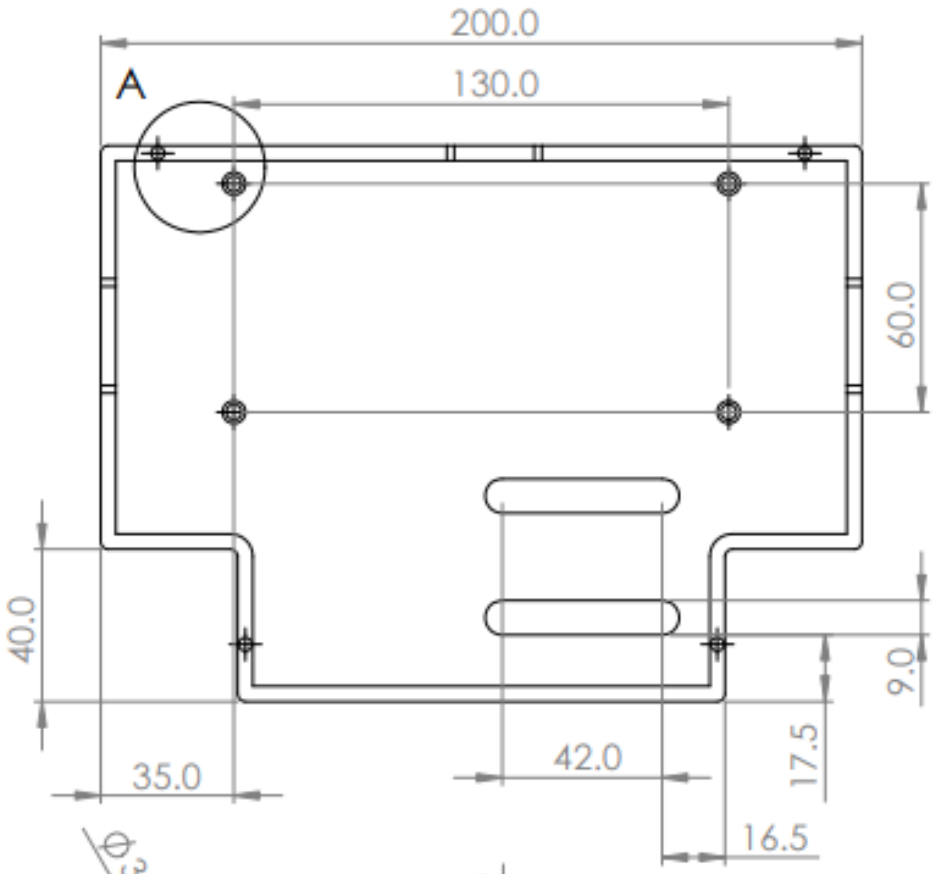
VIEW

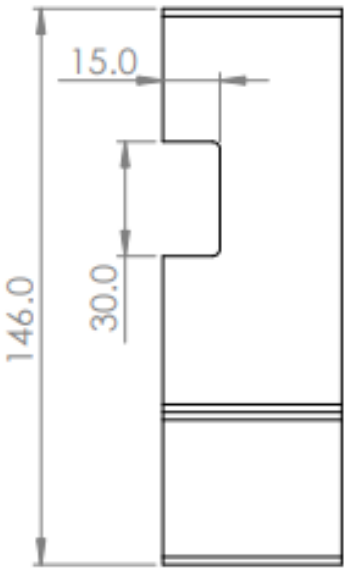


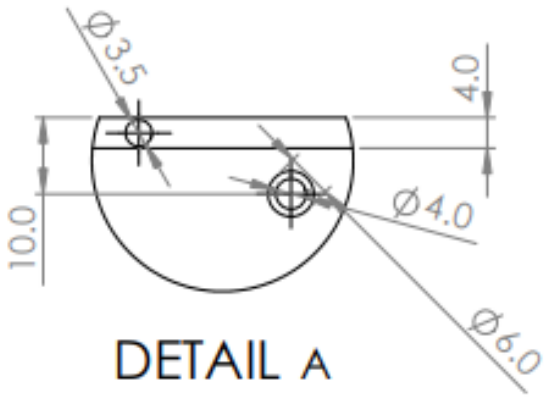
SECTION A-A

UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:	Drawing made in Solidworks 3D	DEBURR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING	DATE: 01/07/2022
			Master Ingeniería Mecatrónica	
UNIVERSITAT POLITÈCNICA DE VALÈNCIA			TITLE: Soporte Bielas	
Project: Vehículo de Ackermann	MATERIAL: PLA		Author Sindre Johannesen	A4
	WEIGHT:		SCALE:2:1	SHEET 1 OF 1



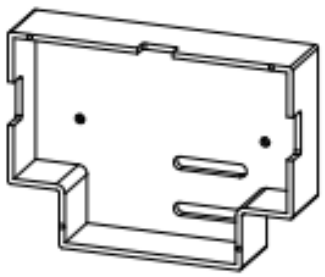






DETAIL A

SCALE 1 : 1



UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:	Drawing made in Solidworks 3D	DEBURR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING	DATE: 01/07/2022
			Master Ingeniería Mecatrónica	
UNIVERSITAT POLITÈCNICA DE VALÈNCIA			TITLE: Carcasa electrónica	
Project: Vehículo de Ackermann	MATERIAL: PLA		Author Sindre Johannesen	A4
	WEIGHT:		SCALE:1:5	SHEET 1 OF 1

Anexo II: Códigos

- Código Matlab
- Código Arduino

Código Matlab que comunica al mando de Xbox con la placa de Arduino

```
% Programa que envia por bluetooth los datos leidos con el mando.
% Ejecutar primero:
% bluetoothlist  --> muestra los dispositivos bluetooth disponibles
% device = bluetooth("PROYEC") --> Se conecta a nuestro modulo HC-05
old_value_a = 0;
old_value_c = 0;

joy = vrjoystick(1)
e = false;
%% Para salir del programa pulsar el boton 'Y' del mando
while ~e
    a = axis(joy,1); % LJ izquierda y derecha
    c = axis(joy,3); % RT (-1) y LT(1)
    e = button(joy,4); % boton Y
    if a ~= old_value_a
        old_value_a = a;
        LJ = uint8(interp1([-1 1], [35,160],a));
        msg = "S" + LJ + "A";
        write(device,msg,"string")
        fprintf(msg + "\n");
    end
    if c ~= old_value_c
        fprintf("C:" + c + "\n");
         if c >  0
             RLT = uint8(interp1([0 1], [127,0],c));
             msg = "M" + RLT + "A";
         else
             RLT = uint8(interp1([-1 0], [255,127],c));
             msg = "M" + RLT + "A";
         end
        write(device,msg,"string")
        fprintf(msg + "\n");
    end
    pause(0.02);
end
```

Código implementado en la placa Arduino que permite el control total del vehículo:

```
/*
  Programa que permite controlar el giro de un servo y un motor de corriente continua
  mediante un mando de Xbox o un smartphone con la aplicacion Bluetooth Electronics
  El mando se conecta al ordenador o  y mediante un programa en Matlab, lee los botones pulsados
  y envian un mensaje por bluetooth al arduino con el siguiente formato:
  Servo: "SXA" siendo X un número entre 50 y 130
  Motor: "MXA" siendo X un numero entre 0 y 255

  O en su defecto mediante la aplicacion que funciona con los mismos mensajes pero añade la
  posibilidad de variar los valores de Kp y KI con los siguientes mensajes:
  Kp: "PXA" siendo X un número entre 0 y 20
  KI: "IXA" siendo X un numero entre 0 y 10
*/
// Libreria que nos permitirá realizar la conexión en serie usando pines distintos a los 0 y 1
#include <SoftwareSerial.h>
#include <string.h>
#include <Servo.h>

Servo myservo;

// Creamos la conexión serie con el móduo BlueTooth
SoftwareSerial BT1(10, 11); // RX | TX

// pines correspondientes de los motores
char PWMR = 13;    // verde oliva
char PWML = 7;     // blanco
char EnaR = 24;    // azul
char EnaL = 28;    // verde

// Pines correspondientes al encoder
#define pinENC_chA 3
#define pinENC_chB 2

// Variables control PI
long t1 = 0 ;
long t2 = 0 ;
int T = 25 ;              // Periodo de muestreo (ms)
float Tm = T/1000.0;      // Periodo de muestreo (s)
float C2S = 1/(374*Tm);   // Conversion de pulsos a cm/s
float referencia;         // Velocidad deseada
float error;              // Velocidad actual
float acc = 0.0 ;         // Accion de control
float vel = 0.0 ;         // Velocidad actual
long quad = 2241 ;
long enc_ant = 0 ;
long enc_act = 0 ;
int enc_dif = 0 ;
float Ik;
float Ik_1;
float Kp = 10;
float KI = 5;
int cont_print = 0;
// Al leer el mensaje recibido, usaremos esta variable para ver sobre que motor(o variable) actuar
char state;
String msg = "";
// Array en el que guardaremos los datos leídos
char received_char[8]; // array en el que guardaremos los datos leídos
const byte numChars = 8;

bool newData = false; // Variable que usaremos para controlar cuando ha entrado datos por el bluetooth y cuando no

int dataNumber = 0;   // Variable en la que guardamos el numero leido por el modulo bluetooth
int ValEnvi = 0;      // Variable ajustada indicando la velocidad
```

```
64
65  void setup()
66  {
67      // sets the pins as outputs:
68      pinMode(EnaR, OUTPUT);
69      pinMode(PWMR, OUTPUT);
70      pinMode(EnaL, OUTPUT);
71      pinMode(PWML, OUTPUT);
72      analogWrite(EnaR, LOW);
73      analogWrite(EnaL, LOW);
74      analogWrite(PWMR, LOW);
75      analogWrite(PWML, LOW);
76      myservo.attach(5);
77      myservo.write(90); // mueve el servo a la posición central
78      Serial.begin(115200);
79      BT1.begin(9600); // Conexion al modulo Bluetooth
80
81      // Interrupciones del encoder
82      attachInterrupt(digitalPinToInterrupt(pinENC_chA),encoder_chA,CHANGE) ;
83      attachInterrupt(digitalPinToInterrupt(pinENC_chB),encoder_chB,CHANGE) ;
84
85      delay(500);
86  }
87  void loop()
88  {
89    //
90    recWithStartEndMarkers();
91    showNewData();
92
93    error = referencia - vel;
94    Ik = error*Tm+Ik_1;
95    acc = -(error *Kp + KI*Ik);
96
97    // Lineas que actuan a modo de AntiWindup
98    if (Ik > 0)
99    {
100     Ik = min(Ik,50);
101   }
102   else if(Ik < 0)
103   {
104     Ik = max(Ik,-50);
105   }
106
107   Ik_1 = Ik;
108   enc_act=quad ;
109   enc_dif=enc_act-enc_ant ;
110   vel=(enc_act-enc_ant)*C2S ;
111   enc_ant=enc_act ;
112   cont_print++;
113
114   // Para poder mostrar adecuadamente los resultados en el smartphone,
115   // Enviaremos los datos cada 10 loops realizados
116   if(cont_print == 10)
117   {
118     BT1.print("*G" + String(referencia) + "," + String(vel) + "*");
119     cont_print = 0;
120   }
```

```
121
122   // Con estas lineas limitamos el valor maximo de la accion de control enviada al motor
123   if (referencia < 0)
124   {
125     if (acc>=255) {acc=255 ;} ; if (acc<=0) {acc=0 ;}
126     digitalWrite(EnaL,HIGH); //forward
127     digitalWrite(EnaR,LOW);
128     analogWrite(PWML,abs(acc));
129     analogWrite(PWMR,0);
130   }
131   else
132   {
133     if (acc>=0) {acc=0 ;} ; if (acc<=-255) {acc=255 ;}
134     digitalWrite(EnaL,LOW);  //reverse
135     digitalWrite(EnaR,HIGH);
136     analogWrite(PWMR,abs(acc));
137     analogWrite(PWML,0);
138   }
139
140   t2=millis() ;
141   if (t2-t1>T) {Serial.println("Error") ;}
142   while (t2-t1<T) {t2=millis() ;}
143   t1=millis() ;
144 }
```

```
// ---------  FUNCIONES ----------

// funcion que se encarga de recibir los datos, usaremos unos marcadores de inicio y
// final para delimitar los datos recibidos
void recWithStartEndMarkers()
{
  static bool recvInProgress = false; // variable que nos indica que empezamos a
                                      //leer datos recibidos por el módulo
  static byte ndx = 0;  //  contador que cuenta los datos que vamos leyendo
  char endMarker = 'A'; // caracter que indica el final de trama
  char rc;  // caracter donde guardaremos el valor leído por el puerto serie

  while (BT1.available() > 0 && newData == false)
  {
    rc = BT1.read(); // guardamos el dato leido por el HC-05

    // Si ya hemos leido el inicio de trama ('R', 'G' o 'B')
    // pasamos a leer el propio mensaje
    if (recvInProgress == true)
    {
      // seguir leyendo hasta que leamos la 'A' que nos indica el final de trama
      if(rc != endMarker)
      {
        received_char[ndx] = rc;
        ndx++;
        // Esto creo que es para que no se desborde si el mensaje tiene mas de 16 caracteres
        if (ndx >= numChars)
        {
          ndx = numChars - 1;
        }
      }
      else // Si hemos leido el final de trama
      {
        received_char[ndx] = '\0';// terminar el string
        recvInProgress = false;
        ndx = 0;
        newData = true;
      }
    }
    // Comprueba la primera letra del mensaje recibido
    // para ver sobre que pin hay que actuar
    else if (rc == 'M')
    {
      state = 1;
      recvInProgress = true;
    }
    else if (rc == 'S')
    {
      state = 2;
      recvInProgress = true;
    }
    else if (rc == 'P')
    {
      state = 3;
      recvInProgress = true;
    }
    else if (rc == 'I')
    {
      state = 4;
      recvInProgress = true;
    }
  }
}
```

```
209  // Funcion que una vez hemos leido los datos del bluetooth nos muestra por el monitor
210  // los datos leidos y escribe el valor correspondiente al pin indicado
211  void showNewData()
212  {
213    if (newData)
214    {
215      // conviene mencionar que el dato recibido vendra en forma de un array de caracteres
216      dataNumber = 0; // que convertiremos a un entero mediante la funcion atoi()
217      dataNumber = atoi(received_char);
218      newData = false;
219
220      switch (state)
221      {
222        case 1:
223          if(dataNumber > 127)
224          {
225            ValEnvi = map(dataNumber,128,255,0,255);
226            referencia = ValEnvi/10;
227          }
228          else
229          {
230            dataNumber = 127 - dataNumber;
231            ValEnvi = map(dataNumber,0,127,0,255);
232            referencia = -1*ValEnvi/10;
233          }
234          break;
235        case 2:
236          dataNumber = map(dataNumber,35,160,160,35);
237          myservo.write(dataNumber);
238          break;
239        case 3:
240          Kp = dataNumber;
241          BT1.print("*P" + String(dataNumber) + "*");
242          break;
243        case 4:
244          KI = dataNumber;
245          BT1.print("*I" + String(dataNumber) + "*");
246          break;
247        default:
248          ;
249      }
250      state = 0;
251    }
252  }
253
254  void encoder_chA (){if(digitalRead(pinENC_chA)==digitalRead(pinENC_chB)){ quad--;} else{quad++;} }
255  void encoder_chB (){if(digitalRead(pinENC_chA)==digitalRead(pinENC_chB)){ quad++;} else{quad--;} }
```

Anexo III: Datasheet

Contenido:

- Motor 12 V con encoder 70:1 Pololu
- Driver BTS7960 Motor 12 V
- Batería Lipo 11.1 V
- Cargador batería B3 Pro
- Servomotor MG996R
- Arduino MEGA 2560
- Módulo Bluetooth HC-05

Motor 12 V con encoder 70:1 Pololu

Dimensions

Size:	37D × 70L mm[1]
Weight:	205 g
Shaft diameter:	6 mm[2]

General specifications

Gear ratio:	70:1
No-load speed @ 12V:	150 rpm
No-load current @ 12V:	0.2 A
Stall current @ 12V:	5.5 A[3]
Stall torque @ 12V:	27 kg·cm[3]
Max output power @ 12V:	10 W[4]
No-load speed @ 6V:	73 rpm[5]
No-load current @ 6V:	0.15 A[5]
Stall current @ 6V:	3.0 A[5]
Stall torque @ 6V:	16 kg·cm[5]
Motor type:	12V

Performance at maximum efficiency

Max efficiency @ 12V:	52 %
Speed at max efficiency:	130 rpm
Torque at max efficiency:	3.2 kg·cm
Current at max efficiency:	0.68 A
Output power at max efficiency:	4.2 W

General specifications

Lead length:	20 cm[6]
Encoders?:	Y
Encoder resolution:	64 CPR

Driver BTS7960 Motor 12 V

Handson Technology

User Guide

BTS7960 High Current 43A H-Bridge Motor Driver

The BTS7960 is a fully integrated high current H bridge module for motor drive applications. Interfacing to a microcontroller is made easy by the integrated driver IC which features logic level inputs, diagnosis with current sense, slew rate adjustment, dead time generation and protection against overtemperature, overvoltage, undervoltage, overcurrent and short circuit. The BTS7960 provides a cost optimized solution for protected high current PWM motor drives with very low board space consumption.

SKU: DRV-1012

Brief Data:

- Input Voltage: 6 ~ 27Vdc.
- Driver: Dual BTS7960 H Bridge Configuration.
- Peak current: 43-Amp.
- PWM capability of up to 25 kHz.
- Control Input Level: 3.3~5V.
- Control Mode: PWM or level
- Working Duty Cycle: 0 ~100%.
- Over-voltage Lock Out.
- Under-voltage Shut Down.
- Board Size (LxWxH): 50mm x 50mm x 43mm.
- Weight: ~66g.

1 www.handsontec.com

Batería Lipo 11.1 V

- Tipo de batería: Batería LiPo
- Voltaje: 11.1 V
- Capacidad: 3500mAh
- Número de celdas: 3S
- Descarga continua: 25C (30C max)
- Tensión máxima por celda: 4.2 V
- Tensión máxima por paquete: 12.6 V
- Velocidad de carga: 1C (3C max)
- Conectores disponibles: XT60 con conector JST para balanceado
- Dimensiones: 136x42x25mm
- Peso: 350 Gramos

Cargador batería B3 Pro

Specification

Input Voltage:	110-240 V AC
Output Current:	3× 700 mA
Display:	3× Bicolor LED
Max. Charging Current:	3× 850 mA
Size:	100 mm * 60mm * 35mm
Weight:	180 g

Servomotor MG996R

Specifications

- Weight: 55 g
- Dimension: 40.7 x 19.7 x 42.9 mm approx.
- Stall torque: 9.4 kgf·cm (4.8 V), 11 kgf·cm (6 V)
- Operating speed: 0.17 s/60º (4.8 V), 0.14 s/60º (6 V)
- Operating voltage: 4.8 V a 7.2 V
- Running Current 500 mA – 900 mA (6V)
- Stall Current 2.5 A (6V)
- Dead band width: 5 µs
- Stable and shock proof double ball bearing design
- Temperature range: 0 ºC – 55 ºC

Arduino MEGA 2560

MICROCONTROLLER	ATmega2560
OPERATING VOLTAGE	5V
INPUT VOLTAGE (RECOMMENDED)	7-12V
INPUT VOLTAGE (LIMIT)	6-20V
DIGITAL I/O PINS	54 (of which 15 provide PWM output)
ANALOG INPUT PINS	16
DC CURRENT PER I/O PIN	20 mA
DC CURRENT FOR 3.3V PIN	50 mA
FLASH MEMORY	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
CLOCK SPEED	16 MHz
LED_BUILTIN	13
LENGTH	101.52 mm
WIDTH	53.3 mm
WEIGHT	37 g

Módulo Bluetooth HC-05

- Bluetooth protocol: Bluetooth Specification **v2.0+EDR** (Enhanced Data Rate)
- Frequency: **2.4GHz ISM band**
- Modulation: **GFSK** (Gaussian Frequency Shift Keying)
- Emission power: ≤4dBm, Class 2
- Sensitivity: ≤-84dBm at 0.1% BER
- Speed: Asynchronous communication: **2.1Mbps (Max) / 160 kbps**, Synchronous communication: **1Mbps/1Mbps**
- Security: Authentication and encryption
- Profiles: Bluetooth serial port
- Supply Voltage: **+3.3V to 6.0 V**
- Supply Current: **30mA**
- Working temperature: -20 ~ +75Centigrade
- Dimension: **26.9mm x 13mm x 2.2 mm**
- HC-05 Bluetooth module follows the IEEE 802.15.1 standardized protocol, through which one can build a wireless Personal Area Network (PAN). It uses frequency-hopping spread spectrum (FHSS) radio technology to send data over the air.

Fuentes

1. Jing-Shan Zhao y Jian S Dai. *Design of an Ackermann-type steering mechanism*. ResearchGate
2. Yan Ding - https://dingyan89.medium.com/three-methods-of-vehicle-lateral-control-pure-pursuit-stanley-and-mpc-db8cc1d32081
3. Adarsh Patnaikm, Manthan Patel, Vibhakar Mohta, Het Shah, Shubh Agrawal, Aditya Rathore, Ritwik Malik, Debashish Chakravarty y Ranjan Bhattacharyaa. *Design and Implementation of Path Trackers for Ackermann Drive based Vehicles*.
4. Gyu-Sang Cho, Jin-Kyung Ryeu. *An Efficient Method to Find a Shortest Path for a Car-Like Robot.*
5. https://es.wikipedia.org/wiki/Curva_de_Lissajous
6. https://es.wikipedia.org/wiki/Lemniscata_de_Bernoulli
7. https://www.boe.es/diario_boe/txt.php?id=BOE-A-2022-3276